(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 833 260 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
01.04.1998 Bulletin 1998/14

(51) Int Cl.[6]: **G06F 17/30**, G06F 17/60

(21) Application number: **97307536.9**

(22) Date of filing: **25.09.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: **27.09.1996 US 722287**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

(72) Inventors:
- **Fin, Tong-Haing**
  Harrison, New York 10528 (US)
- **Fujisaki, Tetsunosuke**
  Armonk, New York 10504 (US)
- **Kobayashi, Makoto**
  Machida, Tokyo (JP)
- **Shinozaki, Masahide**
  Setagaya, Tokyo 242 (JP)

(74) Representative: **Davies, Simon Robert**
I B M
UK Intellectual Property Department
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Computer system with network interface

(57) The web sharing manager 160 of a receiving sharing client 150A receives duplicated events (e.g. browser requests) and messages from the web sharing manager of a source sharing client 150A that causes the browser 130B of the receiving sharing client to execute the duplicate event/message so that the browsers 130A, 130B of the source and receiving sharing client computer system(s) process the same events/messages. Because events/messages, including control and information locations (addresses), are shared between the source and receiving sharing client(s), the same web page is simultaneously displayed and controlled, i.e. shared, on all of the sharing client(s).

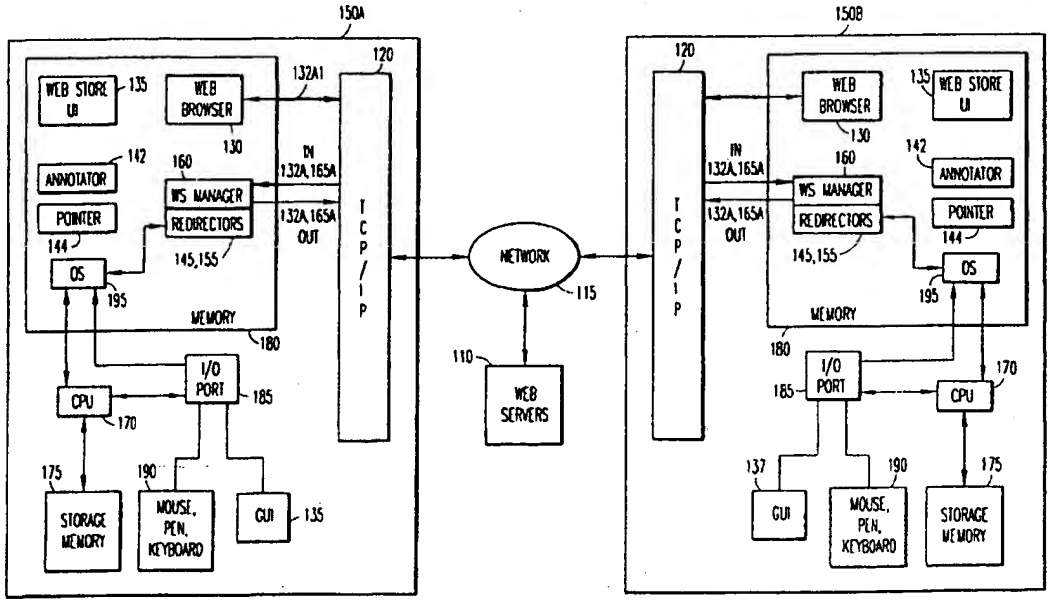

FIG. 1

100

EP 0 833 260 A2

Description

The present invention relates to computers having a network interface, to allow data sharing across the network. The invention is particularly applicable to multiple users sharing the same HTML page on the Internet.

The Internet comprises multiple interconnected computer networks so that any computer on a given network can communicate with one or more computers on any other given network. A gateway computer is a computer which interconnects two networks and passes data from one network to the other(s).

All the computers on the Internet communicate through certain communication protocols, i.e. Internet Protocol (IP). Most applications use Transmission Control Protocol (TCP) along with the Internet Protocol. Therefore the Internet is also referred as a TCP/IP network. Any computer connected to the Internet is identified by a unique Internet address.

The Internet uses a well known packet-switching technology to route information. On the Internet, the data is transferred through packets (also called IP packets) with an address. The speed and reliability of transmission of the IP packets rely on the data traffic and routing. For an example of internetworking communications, see U.S. Patent number 5,371,852 to Attanasio et al. entitled "Method and Apparatus for Making a Cluster of Computers Appear as a Single Host on a Network" issued December 6, 1994.

Some common Internet applications include Electronic Mail, FTP, Telnet and Network news among others. Internet communication protocols and applications are well known.

The World Wide Web (i.e. WWW or Web) is an information service system which is based on the Internet. The WWW uses Hypertext and client/server technology. Hypertext is a method of organizing and presenting information such that any data object in a hypertext file can have links to other hypertext files or data objects. For example, while viewing a hypertext page, a user can select a word which has links. This may bring the user to another hypertext file which contains more text and pictures explaining this word. (The new file may also have more links to other hypertext files and so forth.)

Currently the hypertext file is organized using a HyperText Markup Language (HTML). A hypertext data object can be almost any information medium such as text, image, sound, movie or even a piece of an executable computer program. Any hypertext file on the Web is uniquely identified by its Universal Resource Locator (URL).

The Web client or client (commonly a computer executing a program called a browser) is essentially a hypertext reader which communicates with a Web server through certain data transfer protocols, such as a HyperText Transfer Protocol (HTTP). The client can request a hypertext file with its URL and display the file on a graphical user interface (GUI). This display is called a web page. The client can also pass certain data back to the server and can invoke the Common Gateway Interface (CGI) programs on the server computer to conduct certain tasks. Browsers are commonly known - one popular browser is Netscape Navigator ("Netscape Navigator" is a trademark of the Netscape Communications Corporation).

Using the world wide connections of the Internet, the WWW allows a user, being anywhere on the Internet, to post any hypertext file on the WWW and fetch any hypertext file from all over the world as conveniently as using a local hard disk. This gives the user a tremendous power to acquire information and also turns the Internet into a hypermedia global database, i.e. the information superhighway. The Internet and the WWW have been growing explosively over the past few years. Businesses have also seen the great potential of using the Internet and WWW as the new generation of worldwide communication infrastructure to approach a massive consumer market.

Even though the Web technology gives a user great ability to access/send all kinds of information anywhere in the world, currently users basically have to "travel alone," i.e. users can not present the same information on their web page as other users who are concurrently browsing.

Individual users can run a browser to view web pages. The web pages are in the form of an HTML format. However, the browser used by each individual can not control other browsers in terms of what page another browser is seeing.

Currently, if a first person wants to share a web page with a second user, i.e. "collaborate", the first and second users have to use another communication means, e.g. a telephone, to coordinate their collaboration.

For example, they must coordinate the sharing of the page by sharing the address of the page, selecting entries of the page, communicating movement on the page, communicating movements to and from other locations, etc. This coordination requires that the first and second user exchange a great deal of information very frequently with one another over the other communication means in order to coordinate the browsing.

The prior art does not permit any automatic communication and/or control directly between two or more browsers used by two or more respective users on the internet. Any collaboration, e.g. page sharing, done by the prior art, requires the users to communicate through a communication line other than through the browser. Collaboration requires the facility and the expense of a second communication means. The prior art collaborations often require repeated, and error prone, exchanges of information over the second communication means.

Further, the users can not have "simultaneous collaborations" on a given page. For example, the first user can move to a different location on the page or move to a different page without the second user knowing it. Un-

less the first and second users communicate over a separate communication link, the first user will not be aware of what the second user is seeing/doing on their respective pages, and vice versa. For example, a bank customer looking at a bank HTML home page can not ask a question and/or receive an answer of (from) a bank clerk or agent unless the communication between the customer and the bank agent is done by another communication method, e.g. phone. The agent (customer) does not have control over what page the customer (agent) sees.

The prior art does not permit multiple users to collaborate and/or share information on a common web simultaneously. That is, multiple users do not automatically see the same page (and changes to the page) at the same time. Multiple users also do not share control of the information on the web. For example, they do not see changes at the same time and/or do not see transfers to other pages at the same time.

Accordingly, the invention provides a computer system having one or more memories, one or more central processing units, one or more displays, one or more user input devices, an operating system, and a network interface with a network protocol for communicating to a network, the network connected to one or more servers and one or more shared computers, the computer further comprising:

a browser for requesting one or more pages from one or more of the servers by sending one or more browser requests though the network interface and over the network to the server and the browser receiving from the server one or more of the requested pages sent over the network through the network interface;

one or more redirectors capable of recreating as redirected browser requests one or more of the browser requests necessary to create a web page on the display; and

a web sharing manager connected to the network interface, for receiving the redirected browser requests, and for sending the redirected browser requests over the network to one or more of the shared computers.

In a preferred embodiment, the browser is capable of generating and processing one or more events and said operating system is capable of processing one or more messages, and said system further comprises a queue for stacking one or more queued entries, one or more of the queued entries each containing one of the messages, and one or more of the queued entries each containing one of the events;

and wherein said one or more redirectors are capable of monitoring and recreating as redirected queued entries one or more of the queued entries, and said web sharing manager is capable of receiving the redirected queued events from the redirector and incorporating each of the redirected queued events in a packet conforming to a network protocol which can be sent through the network interface and over the network to one or more of the shared computers.

Typcially, one or more of the messages include any one of the following: an operating system message, an application message, a input device message, and a message produced by one or more of the shared computers; an input device message may be created by one or more of the following: keystrokes from a keyboard, mouse clicks and movements, pen input, or speech signals from a speech recognition system.

Typically one or more of the events is a browser request, more particularly, one or more of the queued events are Common Client Interface (CCI) events created by the browser. In the preferred embodiment, the CCI event is converted into an operating system protocol, preferably Dynamic Data Exchange (DDE).

Examples of CCI events include any one or more of the following: a request for opening a page, a request to change a window size, a request to register a protocol, a request to unregister a protocol, a request to get window information, a request to be notified when a new page is open, and a request to be notified when a window size is changed.

In one preferred embodiment, the input device is a pen capable of producing a signature and the document has one or more windows where the signature is placed.

The invention further provides a method of operating a computer system having a display, a network interface for communicating to a network, and a browser for sending requests for pages over the network and displaying the requested pages on the display, said method including the steps of:

retrieving a message from a memory;

duplicating the message if the message is necessary for a browser to create a page on the display; and

sending the message over a network to a sharing computer so that a receiving browser on the sharing computer can use the message to create a received page on a received display that is identical to the page.

The system and method described herein allow two or more users of the Internet to simultaneously collaborate (view, move on, and/or change) a HTML page, and allow one or more browsers used on the Internet to simultaneously control one or more other browsers used on the Internet. In particular one or more standard browsers can be used on the Internet to simultaneously control one or more other standard browsers used on the Internet without modifying the browsers or computer operating systems.

In a preferred embodiment, one or more client computer systems have a memory, one or more central processing units (CPU), an interface to a network, and

a browser. A network, e.g. the Internet, provides a communication link between the one or more first (e.g. source) clients and one or more second (e.g. receiving) client computer systems using one or more Web servers. The second client(s) also has (have) a memory, a CPU(s), a network interface and a browser. In the first and second clients, the browser is capable of accessing one or more HTML pages from one or more of the Web servers by sending one or more browser requests to any one of the Web servers.

The sharing clients are one or more first and one or more second client computer systems that are collaborating with one another through their respective web sharing manager. The web sharing manager of a receiving sharing client receives the duplicated events (e.g. browser requests) and messages from the web sharing manager of a source sharing client and causes the browser of the receiving sharing client to execute the duplicate event/message so that the browsers of the source and receiving sharing client computer system(s) process the same events/messages. Because events/messages, including control and information locations (addresses), are shared between the source and receiving sharing client(s), the same web page is simultaneously displayed and controlled on all of the sharing client (s). Examples of events that are shared between clients include the following browser requests: location of information on the WWW, gateway connections to additional networks (including telephone networks), etc. Examples of messages include: input/output controls like mouse controls, pen controls, and keyboard input; operating system messages, and application messages.

In a particularly preferred embodiment, there is no need to change the browser or the operating system of the computer from currently available version, as long as the browser has a "registration function" and the operating system has a "hooking function". The registration function causes certain events, for example CCI events, to be routed to a redirector while the hooking function permits a message redirector access to a message queue of the operating system. The registration function and hooking function work with one or more redirectors in the sharing clients to route duplicate browser events and messages among the sharing clients so that the same web page appears on each of the sharing clients.

Preferred embodiments of the invention will now be described in detail by way of example only with reference to the following drawings:

Figure 1 is a block diagram of a preferred embodiment of the invention used in a client/server network architecture;

Figure 1A is a block diagram of a detail of the system in Figure 1 that also shows the flow of events (e.g., browser requests) and messages;

Figure 2 shows a sample of a Web document (web page) that can be shared by the system of Figure 1;

Figure 3 shows a hierarchy of windows (parent-child relationship) for the Web sample document of Figure 2;

Figure 4 shows an example of a shared Web document and the use of mapping;

Figure 5, comprising Figures 5A and 5B, are example block diagrams of a mapping table data structure;

Figure 6 is a block diagram of communication packet;

Figure 7 is a flow chart showing the steps of registering, processing, and unregistering an event (e.g. CCI event) during a web sharing collaboration;

Figure 8 is a flow chart showing the steps creating and sending a packet with an (CCI) event by a source machine;

Figure 9 is a flow chart showing the steps of receiving a packet with an (CCI) event at a receiving machine;

Figure 10 is a flow chart showing the steps of registering a message queue hooking, processing an input message, and unregistering a message queue hooking during a web sharing collaboration;

Figure 11 is a flow chart showing the steps creating and sending a packet with an input message by a source machine;

Figure 12 is a flow chart showing the steps of receiving a packet with a input message at a receiving machine;

Figure 13 is a diagram showing examples of various web page sharing collaborations between a customer(s) and an agent(s); and

Figure 14 is flow chart of process steps performed by a message redirector process.

Figure 1 is a block diagram of one preferred embodiment of the present system 100. The system 100 is a computer network 115 with one or more servers 110 and one or more client computers 150 (e.g., 150A and 150B). In a preferred embodiment, the network uses a TCP/IP protocol 120 for communication among the various clients 150 and servers 110. The protocol 120 is hardware that connects clients 150 to the network 115 and usually also includes software resident on the client main memory 180. The protocol 120 hardware and software (also referred to as the network interface) are well known. One preferred embodiment of the network 115 is the internet and the WWW.

Each of the clients 150 (e.g. 150A and 150B) is a computer with a central processing unit (CPU 170), storage memory(ies) 175, main memory(ies) 180, and input/output (I/O) interface(s) 185. The I/O interfaces 185 permit input devices 190, like a mouse, keyboard, pen overlay, virtual reality inputs, speech recognition system to be processed by the CPU 170 and operating system 195. Other I/O interfaces 185 operate graphical user interfaces (GUI) 135 on the client. Equivalent client computers 150 are well known. One preferred example of a

client computer is any IBM personal computer, e.g. an Aptiva (a trademark of the IBM Corporation). Speech recognition systems like the IBM VoiceType 3.0 dictation system are also well known. (VoiceType is also a trademark of the IBM Corporation). Examples of virtual reality inputs and systems are given in International Patent Application Number WO 95/08793 assigned to Virtual Universe Corporation (U.S. Patent Application number 08/125,950) with a priority date of September 22, 1993.

Each of the clients 150 (e.g., 150A and 150B) participating in this web page sharing also have processes executed by the client. See Figure 1A for more detail of Figure 1. Typically, these processes reside on the main memory 180 of the respective client (150A and 150B) and include a known Web Browser 130, a known Web Sharing User Interface 135, an optional known Annotation device (Annotator) 142, and an optional known Remote Pointer 144. A Redirector (in one preferred embodiment, a Common Client Interface Redirector) 145, a Message Redirector 155, and a Web Sharing Manager 160 are also provided as processes running on each client 150.

In the following discussion, one or more clients originating an event (e.g. a request) and/or message in the web page sharing collaboration is called a source or sending sharing client (or machine) or source client/machine and is designated 150A. Reciprocally, one or more sharing clients receiving an event and/or message is called a receiving sharing client (or machine) or receiving client/machine and is designated as 150B. Of course, these sharing clients can and do interchange roles. Components of the source 150A and receiving 150B machines have their numeric designators appended with "A" or "B", respectively.

For clarity in Figure 1, some processes are omitted. However, more detail is shown in Figure 1A. All clients 150 have the Web Browser 130, Web Sharing User Interface 135, a redirector (e.g. a Common Client Interface (CCI) redirector 145), message redirector 155, and web sharing manager 160. Clients 150 also typically have input/output (I/O) ports 185 and optional drivers like the annotation (annotator) 142 and remote pointer 144 drivers.

In a preferred embodiment, the web sharing manager 160 and redirectors (145, 155) are processes that are independent of the browser 130 and operating system 195. However, the redirector processes (145, 155) connect to the browser 130 through a registration function 130F and to the operating system 195 through a hook 191 so that they can access event information from the browser and access and provide message and event information stored in a memory 165 used by the operating system 195. In a preferred embodiment, the memory 165 used by the operating system to store and access messages and request/event information (in a message form) is a message/event queue (or queue) 165. The redirectors (145, 155) make sure that the request/events and messages are routed to the correct place in both the source and receiving sharing clients so that the same web page appears on all sharing clients.

The structure of the queue 165 is well known and defined by the operating system. Typically, the queue entries 171 (messages and events/requests in "DDE" message form) in the queue 165 have the structure including: a message type 166, parameters 167 associated with the message, a destination 168 which specifies the receiver of the message, and other information 169.

By loading the web sharing manager 160 and the redirectors (145, 155) into the memory 180 of the sharing client 150, any client with a standard browser 130 (having a registration function 130F) and a standard operating system 195 with a hook 191 to the message/event queue 165 can be converted to a sharing client 150 that can collaborate with other sharing client(s) 150 without modification to the browser 130 or operating system 195.

In a preferred embodiment, the computer system 150 uses the redirector (specifically a CCI redirector) 145 to send duplicate (CCI) events 132AOut, generated by the browser 130 of the source client 150A, to one or more receiving clients 150B. The (CCI) redirector 145 of the receiving clients 150B causes received (CCI) events 132AIn to be processed appropriately and routed to the browser 130 in the receiving client 150B for execution. (Note that in this disclosure, event and CCI event and redirector 145 and CCI redirector 145 will be used interchangably without loss of generality.)

The sharing client computer system 150 also uses the message redirector 155 to duplicate messages (like keyboard and mouse inputs 185) produced by the source client 150A and sends these duplicate messages 165AOut to the receiving client(s) 150B. The message redirector 155 is used by the receiving client 150B to mark or tag (see below) received messages 165AIn and works with the operating system 195 to process these received messages 165AIn.

The Web Browser 130 can be any commercially available network browser that has a registration function, e.g., Netscape Navigator 130. The Web Browser 130 communicates to the network 115 and servers 110 using well known internet protocols (e.g. http, ftp) on top of a TCP/IP protocol.

The Web Sharing User Interface 135 provides the look and feel for users of this package to establish the connection, to invoke tools (for example, remote pointer 144, annotation 142), to share or unshare the Web browser's document, and to show the status of other users. Since the user interface component 135 used in this system 100 is a simple and known function that can be found on most desktop conferencing systems, it will not be described further.

The annotation or annotator 142 provides handwriting (or pen based) capability to write comments or highlight some areas of a web document and the remote

pointer 144 is used to point to or to move around the objects on a shared document. A pen input device 190, if it is available, may be used for annotation or pointing (alternatively, a mouse 190 can be used). This capability of annotation 142 and the remote pointer 144 is well known in the desktop video conferencing products (see for example, the IBM Person to Person product).

Refer again to Figure 1A, a block diagram of a detail of the system in Figure 1, that shows the flow of information and controls (events and messages) among components in the system.

A preferred embodiment includes two software components: the CCI redirector 145 and the message redirector 155. The CCI redirector 145 monitors and routes the CCI events 132A that the browser 130 sends and/or requests and the CCI events 132AIn that are received from the network 115. The message redirector 155 captures and routes the input data messages from input devices (e.g., keyboard, mouse, pen messages) 190 and other sources. In alternative embodiments one "redirector" could perform the functions of both the CCI redirector 145 and the message redirector 155. In other alternative embodiments, the redirecting could be performed by the browser and/or the web sharing manager that are designed specifically to include these functions.

The preferred embodiment uses a "registration" feature 130F of known browsers 130. Many browsers call this registration feature a Common Client Interface (CCI) (132, 130F). The CCI 132 of some Web browsers 130, for example the NetScape Navigator, provide features for external programs to control and extend the functionalities of the browser 130 through this cross-platform interface (CCI 132). The CCI 132 is specified as a set of interfaces which can be used to communicate either to or from Web browsers 130. Using this CCI 132, the user defines the CCI events 132A that the browser is to pass to an external process, here the CCI redirector 145. Therefore, every time that the browser 130 generates a "registered" CCI event, the browser not only sends the CCI event 132A1 through the TCP/IP protocol 120 to the network 115 but also sends a duplicate CCI event 132A to the operating system 195. In a preferred embodiment the operating system 195 applies a Dynamic Data Exchange (DDE) protocol 131 to the duplicate CCI event 132A before the operating system 195 stores the CCI event 132A, preferably in the queue 165. The DDE protocol makes the CCI event/request 132A appear as a DDE message in the queue 165. The detail specification of CCI 132 for the NetScape Navigator is described in http://home.netscape.com/newsref/std/ddeapi.html for the Netscape browser 130.

The Netscape Navigator in a Windows platform uses DDE 131 to implement CCI 132 (Windows is a trademark of Microsoft Corporation). DDE 131 is a known message based interprocess communication protocol defined in the Windows platform and used by the browser 130 and as described herein by the CCI 132.

For example, when a user/client 150 opens a new Web document by explicitly specifying a URL of the document or by clicking a hyperlink to a URL document, the browser 130 sends an event (request) 132A1 requesting a page from the Web server 110 through the TCP/IP interface 120. Since the CCI redirector is pre-registered (130F, 132) for monitoring CCI events (see Figure 7 below), the browser 130 also sends a (duplicate) CCI event 132A called OpenURL to the operating system 195 that applies the DDE protocol 131 to the event 132A and places it one the queue as a DDE message. The CCI event 132A is eventually routed (see discussion and Figure 14 below) to the CCI redirector 145 which causes the CCI event 132A to be routed through the web sharing manager 160 (as a "packet" -see Figure 6 below) and through the network interface 120 over the network 115 to the shared receiving clients 150B.

As another example, when the window of the Web browser 130 is changed (e.g. resized or maximized by a user 150), a CCI event 132A called WindowChange is sent by the browser 130 to the CCI redirector.

In DDE protocol terminology, OpenURL or WindowChange CCI events 132A are set in a DDE topic name and the URL of the document or the new size of the browser's window is set in a DDE item. In the entry 171 of this DDE message, the message type 166 will be set to "DDE message", the parameter field 167 will contain CCI event 132A, and the destination field 168 will be the CCI redirector.

The message redirector 155 routes all messages incoming 165AIn from the network 115 to the queue 165 and monitors all queue entries (events/requests and messages) on the queue 165 that are processed by the operation system 195. The message redirector 155 monitors the queue 165 by "hooking" the queue. The message redirector 155 processes certain queue entries, e.g. messages and events, in the queue 165 in which the message redirector 155 has an "interest". This is described in more detail below.

The preferred embodiment uses a known feature called hooking 191 provided by the operating system 195 to catch all queue entries 171 in the queue 165. If a hook is installed with the operating system 195, all retrieved queue entries 171 from the queue 165 are passed to the hook 191. The hook 191 permits the message redirector 155 to check 156 the message type 166 to determine whether the message redirector has an "interest", i.e. will process the queue entry 171. If the message redirector 155 has no interest, control is passed back to the operating system 195 so that the queue entry 171 can be further processed by the operating system 191. However, if the message redirector 155 has an interest 156, the message redirector 155 processes the queue entry before the operating system is allowed to deliver the queue entry 171 to its destination, e.g. an application. After the message redirector 155 has completed its processing, control is returned to the operating system 195 to further process the queue entry 171. The message redirector 155 has an interest in those mes-

sages (not including DDE messages handled by the CCI redirector) necessary to recreate the web page at the source sharing client 150A on the receiving sharing client(s) 150B.

For example, the CCI event 132A is put into the queue 165 by the operating system 195 as a DDE type 166 message. When the operating system 195 is requested to retrieve a queue entry (the DDE message/CCI event 132A), the operating system 195 checks if there is any hook 191 before delivering this CCI event 132A to the destination application. Since the message redirector 155 hooks 191 the queue, the operating system 195 passes control of the CCI event 132A to the message redirector 155. The message redirector 155 checks the message type field 166 to determine 156 if the message redirector 155 needs to process the queue entry 171. Since, in this embodiment, the message redirector 155 ignores all CCI events (DDE message types 166), the processing of the CCI event 132A by message redirector ends and control passes back to the operating system 195.

The operating system 195 then uses the destination field 168 of the DDE message to deliver 193 the DDE message. If the CCI event originated in the browser 130, the browser placed the CCI redirector 145 as the destination in the destination field 168. However, if the CCI event 132AIn originated from an external source sharing client and was received from the network, the CCI redirector 145 placed the browser 130 (associated with its client 150A) as the destination in the destination field 168. In either case, these CCI events are made into DDE messages by the DDE 131 and placed on the queue 165.

As the entries 171 in the queue 165 are processed (see hook 191 below) the operating system determines how to route the CCI events 193. In the case of the CCI event that was received from a source sharing client 150A, the destination field 168 indicates that the message will be directed 193 to the browser 130. However, the CCI events that are generated at the source sharing client 150A have a destination 168 of the CCI redirector 145 and are directed 193 there 145. The CCI redirector requests the Web Sharing Manager 160 to send this CCI event 132AOut across the network 115 to the receiving sharing client 150B. The receiving sharing client 150B will use this CCI event (now a 132AIn) to duplicate the CCI event (e.g. request) on the receiving sharing client 150B so that the web pages on all of the shared clients are kept the same.

A "hook" 191 is a well known operating system 195 defined function that allows external processes to intercept queue entries 171 in the memory, e.g. the queue 165, of the operating system 195. Once an external process invokes the hook, the operating system 195 passes requested information, e.g. one of the queue entries 171, to the external process.

Once the external process, e.g. the message redirector 155, completes its function, the external process passes control back to the operating system 195 by some indication like a "return" in the function call. For example, in the Windows 95 operating system the "hook" is defined in the "Software Development Kit" (SDK) referenced in Win32 Programmer's References. For the OS2 operating system (OS/2 is a trademark of IBM Corporation), it is defined in Presentation Manager Programming References.

Queue entries 171 can originate from several places. As described above, duplicate CCI events 132A originating from the CCI 132 in the browser 130 are placed in the queue 165 after the DDE protocol is applied. Also, received CCI events 132AIn are identified 161 by the web sharing manager 160 and pass through the redirector 145 and DDE 131 to the queue 165. The DDE assigns each of the CCI events (132A and 132AIn) with a message type 166 "DDE message" and a destination 168 as described above.

In addition to events, queue entries 171 include messages. Messages can be generated by the operating system 195 (internal messages) or by sources external to the operating system 195 (external messages). External messages include messages from the input devices 190 and the I/O port 185, messages generated by inter-process communications, and messages received from a sharing client 150 connected through the network 115.

The operating system 195 generates internal messages that include: "scrolling", "move a window", "resize a window", "minimize a window", and "maximize a window". These messages are defined in the SDK for the operating system 195. The operating system 195 gives these internal messages a "window message" message type 166 and the window handle of the appropriate window as a destination 168.

Messages from input devices 190 include: keyboard messages like "key up", "key down", and "key code" (e.g. an ASCII value); mouse messages like "mouse up", "mouse down", "mouse move", and "mouse double click"; pen messages like "pen up", "pen down", and "pen move"; and annotator messages line "annotator up", "annotator down", and "annotator move." For each of these messages, the operating system 195 assigns a queue entry 171 message type 166. For example, if the left mouse button moves up, the message type 166 will be "mouse left button up." Further the destination field 168 will contain the window handle of the window where cursor is located. The parameter field 167 will contain the (X,Y) of the mouse cursor and the other field 169 might contain other information like a time stamp. Populating these queue entry data structures 171 in this way is well known.

Messages are commonly used as the inter-process communication between applications, especially in a message based window system (e.g. Windows 3.1 or OS/2). When an application requests an operating system 195 to send a message to another application, the operating system 195 puts the message into a queue

165 and then delivers the message when the destination application requests the operating system 195 to retrieve a message from the queue 165. This message event/queue 165 is also used to put and deliver other known messages to the applications (e.g. when the window of the application needs to be repainted).

Similar to the inter-process communication between applications, communications between the input devices 190 and the applications are also through the queue 165. All input data, defined above by example, from input devices 190 (e.g. keyboard, mouse, pen) are transformed into well defined messages by the window operating system 195 and are put into queue 165. To read its input data, each application will request the window system to retrieve the messages from the message event queue 165 as a queue entry 171.

In one preferred embodiment, messages are received by one or more receiving sharing clients 150B from the network 115. These messages are passed through the web sharing manager 160 which determines 161 whether the received signal is a message or an (CCI) event (e.g. request). If the received signal is a message 165AIn, the message is passed through the message redirector 155 and then to the operating system where it is placed on the queue just as any other external message. However, in this case, the message 165AIn is tagged to indicate that it came from the message redirector 155, i.e., the network 115. In a preferred embodiment, the message redirector 155 tags the message by adding a fixed number (e.g. an offset) to the message type field 166 which produces a message type 166 that is understood by the message redirector to identify any message from a remote source sharing client 150A. Note that these messages 165AIn can include any of those messages that are generated by a source sharing client 150A and received over the network 115 either as application messages, input device messages, or as internal operating system messages.

Since the message redirector 155 has hooked 191 the queue 165, the message redirector 155 examines each of the queue entries 171 in the queue 165. However, the message redirector 155 only has an interest, i.e. only processes, certain of these queue entries 171. In general, the message redirector 155 needs to processes only those queue entries 170 that are required to create the same page on any of the sharing clients 150.

Examples of queue entries 171 that are processed by the message redirector 155 include queue entries with message types 166: "key code" for the key board, "mouse (left or right) button (click or double click)", "mouse move", "scrolling", and "window (get/lost input focus)" messages. These messages are tagged by the message redirector.

Examples of queue entries 171 that typically are not processed by the message redirector 155 include queue entries 171 with message types 166: "key up (down)", "mouse (left or right) button (up or down)", "window (move or resize)", "window (active or inactive)", "window (create or destroy)", "menu selection", and any DDE message. This is because these queue entries do not effect the appearance of the web pages displayed by the browser.

As the operating system 195 processes the queue entries 171, "non DDE" messages are hooked 191 to the message redirector 155. Upon determining 154 that the message redirector 155 has an interest in the message, control is passed to the message redirector 155 and the queue entry 171 is processed by the message redirector 155. If the message redirector recognizes the message type 166 of the message as that indicating the message came from a source sharing client 150A, the message redirector changes the message type 166 back by removing the tag, e.g., by subtracting the offset from the message type 166. The message redirector 155 also changes the destination 168 of the message to that of the browser 130, replaces the message in the queue, and returns control to the operating system 195. These messages, even though they originated at a remote source sharing client 150A are processed by the browser 130 to maintain the web pages of the sharing clients 150A,B in the same state.

However, if the queue entry 171 is a message that originated at the source sharing client 150A, the message redirector duplicates the message 165AOut and sends it through the web sharing manager 160 and the network interface 120 over the network 115 to the receiving sharing clients 150B.

Control is then returned to the operating system 195 which processes the message within the source sharing client 150A.

The Web Sharing Manager 160 provides the functionality to send both the CCI events 132AOut and messages 165AOut through the TCP/IP network interface 120 to the receiving sharing client(s) 150B. Also, the Web Sharing Manager 160 creates a proper header for receiving packets, through the network 115, from source sharing clients 150A. This is described in more detail with reference to Figure 6 below.

The CCI events 132AIn and messages 165AIn sent to the sharing receiving client 150B inform the web sharing manager(s) 160B of the sharing receiving client 150B what the source sharing client 150A is doing. The web sharing manager 160B receives the sent CCI events 132AIn and messages 165AIn, strips the Web Sharing Manager header 620 of Figure 6, and determines 161 whether they are CCI events or messages in order to route them to the CCI redirector 145 or message redirector 155, respectively. The operating system 195, message redirector 155, and CCI redirector 145 then operate together, as defined above, to permit the real time sharing or collaboration, i.e. all sharing clients 150A,B view and control their respective web page the same way. Also the same information, e.g. data, messages, etc. is viewed on the GUI displays (web pages) of all the sharing clients.

By using this technique, all sharing clients 150A,B

collaborate to share the same web page, i.e. both or all users will see the same Web document displayed by their respective Web browser 130. If a sending user 150A, for example, inputs some keystrokes from the keyboard, creating message events 165A, the same keystrokes will be displayed on the receiving user 150B screen in the same way as if these keystrokes are input by the receiving user 150B. This sharing function of input is realized by the message redirector 155 which captures the input messages (e.g., keyboard, mouse messages) in the message queue 165 in the source client 150A and redirects the same messages as message events 165A to be replayed in receiving sharing clients 150B.

All the processes performed on the clients 150 are transparent to the Web server(s) 110. The web server only handles communication packets (see Figure 6 below) with standard communication headers 610 that contain the sharing requests and information. All other functions of the server are the same as any typical WWW function. In a typical environment, the Web document(s) displayed by the Web browser 130 is (are) retrieved from the Web server(s) 110. With the hyperlink capability, this Web document may be linked to other Web documents in the same Web server 110 or in other Web servers 110 in the network 115. The Web server 110 is used as the repository of Web documents browsed by the Web browser 130.

Therefore, the CCI 132 with the CCI redirector 145 (and the message redirector 155) monitor CCI events 132A (messages 165A) independent of the use of these CCI events 132A (messages 165A) by the system 150A, so that they can be transparently passed over the network 115 to other sharing receiving clients 150B collaborating with the source sharing system(s) 150A.

Figure 2 shows an example of a Web document 200, displayed on the GUI 135 of the Web browser 130, which includes, for instance, a parent window 200 and one or more child windows (201-205). Here the child windows include: one text entry 201 object (e.g. for entering an account number), one multiple-line entry object 202 (e.g., for comment), two push-button objects (e. g. for submitting 204 and clearing 205 the document), and a novel white area 203 (e.g. for a pen based or pointer based signature.) The Web document 200 is shown in the browser's window and other objects are created as the child windows (201-205) of this window. By known techniques, the browser window is created when the browser starts and it persists until the browser is ended. The child windows (201-205) are created at the time the browser displays the Web document and they will be destroyed when the browser displays a new document, i.e. the child windows (201-205) in the document 200 are dynamically created and destroyed. Also in Figure 2, the pen 250 and the pointer 260 represent the functionality of the Annotation 142 and the Remote Pointer 144, respectively. When the window of Web document 200 and its child windows (201-205) are created or destroyed, "window (create or destroy)" messages are generated by the operating system 195. However, these messages need not be processed or duplicated by the message redirector 155 since the same messages will be produced in other sharing receiving clients 150B by the operating system 195 of those 150B client (s). This occurs in two instances: 1) moving on to another page and 2) explicitly closing the page. When moving on, the sharing client must create a new page and therefore closes the current page. When explicitly closing a page, the CCI event closing the page is shared with the client 150B and will cause the client's operating system 195B to close the current page.

Figure 3 shows the hierarchy of windows 300 (parent-child relationship) for the Web sample document of Figure 2 embodied in a data structure. In a window system (e.g., Windows 3.1 or OS/2), each window, including the browser window, is identified by a number called a window handle which is unique and is only valid in the window system that creates it. This window handle, or "handle" is used to send/receive messages to/from other windows. The handle for the browser's window 310 is, for example, 2000. The browser window handle 2000 is static and is not changed until the browser is finished running. However, the handles 2021, 2022, 2023, 2024 and 2025 of the respective child windows 321, 322, 323, 324 and 325 will be dynamically created. These child handles (2021-2025) are valid only when the child windows are valid and will not be reused after these child windows are destroyed, e.g. when the browser jumps to a new document. These window handles (2021-2025) are used by this Web sharing system 100 to display the same document and to support the same input from various input devices (e.g. keyboard, mouse, pen) over two or more clients 150.

Figure 4 shows an example of sharing a Web document 200, such as http://www.bank.com/homepage.htm, between a customer (here, for example, the source 150A) and a bank agent (here, for example, the receiver 150B).

There a two types of information to be shared between the customer and the bank agent: the view of the Web document 200 and the input data (e.g. keyboard input for the account number) 233. The sharing of the view of Web document 200 is realized by the CCI redirector 145 as a CCI event 132A, whereas the sharing of input data 233 is realized by the message redirector 155 as a message event 165A. The detailed mechanisms used by the CCI redirector 145 and the message redirector 155 to support sharing of Web documents will be described below.

More detail is now given about the mechanism used by the Web sharing system 100 to share the same Web document between source 150A and receiving 150B machines.

At the source 150A machine, when the customer opens a new URL, the Web document 200(A) is shown in the browser's 130A window having, for example, 2000

as the window handle. After the Web page is displayed, the browser 130A sends a CCI OpenURL event using the DDE protocol to CCI Redirector 145A as notification that the Web page is successfully opened and displayed. The CCI Redirector 145A redirects this OpenURL event 132A to the Web sharing manager 160A to process this event 132A. After receiving this OpenURL event, the Web sharing manager 160 enumerates the hierarchy of windows (as shown in Figure 3) of the open document, assigns an ordering number to each window, and builds a table, for example, as shown in Figure 5A for the customer machine 150A. For example, in Figure 5A, the document 200A of Figure 4 shown in the customer machine 150A has a window handle of 2000 and an ordering number 0. The child window 201A for document 200A for the account number 233A has a window handle 2021 (410A) and an ordering number 1. The window handles are created by the window system and are destroyed each time the associated windows are closed. Therefore, the window handles cannot be used to identify the child windows in the open Web page since the same Web page may be reloaded (closed and reopened) by the browser. Instead of using the window handles, the Web sharing manager 160A uses the ordering numbers to identify the Web document's window and its child windows. For example, the ordering number 0 is used to identify the Web document's window 200A, the ordering number 1 is for the the child window 201A for account number 233A.

At the bank agent (receiving) machine 150B, the Web sharing manager 160B receives the OpenURL CCI event 132A (from the Web sharing manager 160A) through the network 115 and passes this OpenURL CCI event 132A through the CCI redirector 145B to the Web browser 130B for opening a new URL. After receiving this request, the Web browser 130B opens and displays the Web document in its window having, let say, 5000 window handle. Similar to the creation of Figure 5A in the customer machine 150A, the Web sharing manager 160B enumerates the hierarchy of windows (as shown in Figure 3) of the open document 200B, assigns an ordering number to each window, and builds a table, for example, as shown in Figure 5B for the customer machine 150B.

The mapping table data structure 500 is used by the web sharing manager 160 to map the window of the Web browser and its child windows on one client 150A to their corresponding windows on receiving clients 150B.

This mapping table 500 has records, typically 510, each having three or more fields. The first field 520 contains an ordering number of a window, a second field 530 contains the window handle of the respective window, and a third field 540 contains a type of the respective child window (321-325) (e.g. text, bit map, etc.) in the document 200. For instance, this table 500 is used to map a child window 203A in one machine (e.g. 150A) to the corresponding child window 203B in another machine 150B.

Since the browsing of Web document 200 and the processing of input data from the input device 190 are based on the information of the windows, the mapping table, as shown in Figure 5, will allow the Web sharing system 100 to display the same Web document in the correct window of the Web browser 130 and to redirect the same input data to the right Web document.

For example, by using the table 500, all keyboard messages 165A sent to the window handle 2021 in the customer machine 150A are mapped to the messages 165B sent to window number 3 (in the ordering number field 520 of table 500) of the second machine 150B. This window number 3 will be known to have the window handle 5010 in the second machine 150B, e.g. the bank agent machine. Using this mapping mechanism 500, the keyboard input 190 to the text entry window 203A will be sent to the corresponding text entry window 203B, and the customer and the agent will see the same account number displayed on their screen. Other window types are mapped in the same way.

This capability enables all clients 150 to see or share the same Web document shown on their screen by the Web browser 130 and the same input data from input devices 190 into the shared Web document. When a user clicks on the hyperlink to browse a new Web document, other users will automatically follow the hyperlink to the same Web document.

In addition to the sharing of Web document and input data, for example to imitate a face-to-face meeting, one user may use the remote pointer 144 to point to or to move around the objects in the shared document and other users engaged in the Web collaboration session will see the cursor of the remote pointer 144 displayed at and moved around the objects in discussion (since the movement of the remote pointer 144 will generate the mouse input data which are shared as described above). With the annotation 142, a user may use the pen input device, if it is available, or the mouse to write some comments or to highlight some areas on the shared document. The annotated comments or the highlights will simultaneous appear on other clients (since the annotation 142 will generate the pen/mouse input data which are shared as described above). This sharing of Web document and input devices will allow users to have a conference or meeting over the network. Thus, with the aid of the annotation 142 and remote pointer 144, the conference or meeting created by the Web sharing system 100 has the same atmosphere as a face-to-face meeting.

Figure 6 is a block diagram of communication packet 600 with a (e.g. TCP/IP) communication protocol used by the web sharing manager 160 to exchange requests and information between the client machines 150. The packet 600 has a communication header 610, two or more Web Sharing Manager headers 620, and a data section 630 with one or more fields. The first header 610, the communication header, is defined by TCP/IP protocols. This header 610 is well known. Since the Web

Sharing Manager provides common functions to send and receive data for other components (CCI redirector, message redirector, etc.), it uses a special web sharing manager header 620 to identify the receiver (e.g., annotation, remote pointer 144, CCI redirector 145, message redirector 155, web sharing manager 160) of the packet 600. This receiver information is contained in a receiver ID field 622 of the header 620. The web sharing manager header 620 further comprises a packet length 624 field that contains the size (length) of the packet, i. e. the portion comprising the web sharing manager header 620 and the web sharing manager data 630. Other information can be included in the web sharing manager header 620.

The Web sharing data 630 section of the packet 600 is the data to be sent or received over the network 115 to the "receiver" named in the Receiver ID field 622. The web sharing manager strips the web sharing manager header 620 before this packet data 630 is passed to receiver. For example, the packet data 630 for CCI redirector 145 will include the type 632 of CCI event (e.g., Open URL event) and the parameter 634 of the CCI event (e.g. http://www.ibm.com/home.htm URL). Examples of packet data 630 for the receiver's message redirector 155, remote pointer 144, annotation (e.g. pen based input) 144, will include the message types 632: keystroke from the keyboard 190, mouse 190, and pen stroke 190, respectively. The parameters 634 would be keycode, mouse position, and pen position, respectively.

When the Web document 200 displayed by the Web browser 130 is shared, the view of the Web document 200 in the Web browser's window as well as the input data (e.g. keyboard/mouse/pen input data) are shared between the source and receiving machines 150. The sharing of the view of a Web document is realized by the CCI redirector 145 by communicating CCI events 132A whereas the sharing of input data is realized by the Message redirector 155 by communicating message events 165A.

The CCI redirector 145 uses the CCI events (e.g. OpenURL, WindowChange) provided in the Web Browser 130 to implement sharing of the view of the web document 200 as described below. When user 150A opens a new HTML document or browses the documents through a hyperlink, the Web browser 130A at the source machine (e.g. 150A) notifies the CCI redirector 145A at the source machine 150A of CCI events 132A to be sent by the source machine 150A. Also, the CCI redirector 145B at the receiving machine 150B will pass the CCI events 132A at the receiving machine 150B to its respective Web browser 130B for execution in the same way as if these CCI events 132A were generated (as CCI events 132B) from the operations of user 150B. An example of a received CCI event is a request to display a Web document 200 that the sending user 150A opens.

Figure 7 is a flowchart of a process 700, executing on both the sending 150A and receiving 150B machines, showing the steps of initialization, processing a CCI event 132, and a termination done by CCI redirector 145 during the Web collaboration session. At the beginning of the Web collaboration, the CCI redirector 145 registers with the Web browser 130 all the CCI events 132 for which the CCI redirector 145 has an interest. By this registration relationship, step 710, the Web browser 130 will notify the CCI redirector 145 by sending registered CCI events 132 using the DDE protocol 131. Examples of CCI events for which the CCI redirector 145 registers include: RegisterProtocol, RegisterWindowChange, etc. The redirector 145 is interested in any CCI event 132A that is needed to create a document view on the receiving machine 150B.

For example, when a user 150A opens a new URL document, a CCI event 132 called Open URL, which is registered with the browser 130A in step 710 by CCI redirector 145A, is generated by the Web browser 130A and is sent to the CCI redirector 145A using the DDE protocol. By receiving this OpenURL CCI event, the CCI redirector 145A knows that a new Web document 200 is displayed by the Web browser 130A. By sending the name of the open Web document from the CCI redirector 145A (a sender of CCI event) to the corresponding CCI redirector 145B (a receiver of CCI event) through the respective Web sharing managers 160, the same Web document 200 can be displayed on the receiving machine 150B by sending an OpenURL event from the CCI redirector 145B to the Web browser 130B using the DDE protocol 131B.

In step 720, CCI event 132A(B) is processed by CCI redirector 145B(A) as a sender (or a receiver) of CCI event 132A(B). The machine 150A is a sender when the browser 130A sends a CCI event 132A to the CCI redirector 145A and the machine 150A is a receiver when when the web sharing manager 160A passes the communication packet 600 to the CCI redirector 145A. This step 720 is shown in more detail for a CCI sender 145A in Figure 8 and for a CCI receiver 145B in Figure 9. When the Web collaboration is terminated, the CCI redirector 145, in step 730, will unregister all registered CCI events 132 from the Web browser 130.

Figure 8 is a flowchart of the steps 720 implemented on the source or sending client 150A that is notified of a CCI event from the Web browser 130. The CCI redirector 145A first determines 810 that the machine 150 is the source/sender 150A of a request from its web browser 130A. When the source web browser 130A notifies the CCI redirector 145A of a CCI event 132A, step 810, the CCI redirector 145A stores and updates 820 the information (see Figure 5A) related to this CCI event 132A for later use. The source CCI redirector 145A enters the type 632 and parameter 634 of the CCI event 132A into the network web sharing data packet 830. The CCI redirector 145A sends 845 the data packet 630 to its source Web Sharing Manager 160A. The source Web Sharing Manager 160A then creates 850 the entire com-

munications packet 600 by entering the receiver ID 622 and packet length 624 in the Web Sharing Manager header 620 and providing any other communication header 610 information required. Note that the Web Sharing Manager 160 can send packets 600 to more than one client to create a multiple user session by providing a receiver ID 622 and any communication header (TCP/IP) 610 information required for each respective client receiver 150B. The packet is then sent through the interface 120 by the Web Sharing Manager 160A using known techniques, e.g. TCP/IP. Step 840 terminates the CCI process 720 for the source machine 150A for any given CCI event 132A.

Figure 9 is a flowchart of the steps 720 implemented on the receiving client 150B that receives a CCI event 132A from the Web Sharing Manager 160B. The Web Sharing Manager 160B disassembles the Communication Packet 600. The CCI Redirector 145B first determines that the machine 150 is the receiver 150B of a Web sharing data packet from the Receiver ID 622 that was disassembled by the Web Sharing Manager 160B. (Note that the receiver ID 622 can be the CCI redirector 145B or the Message redirector 155B.) After determining the type 632 and parameter 634 of the received CCI event 132A, the CCI redirector 145B converts 910 the CCI parameters to the local ones, using the DDE protocol 131B. This is done by creating and providing the information in table 550. (Note that the DDE protocol of the sender 131A usually is the same as that of the receiver 131B. However, once the CCI event 132A is recreated at the receiver, it can then be converted by the CCI redirector 145B into a new format, e.g. Netscape to Mosaic.)

As an example, in step 910, when machine 150B receives an OpenURL CCI event 132A, the CCI redirector 145B passes the OpenURL CCI event to the browser 130B (the request to execute the event in step 920) which will access the document to be opened from the server specified in the parameter 634. Then, as the browser 130B builds the document view, the CCI redirector 145B determines how many child windows there are and completes the information require in table 550. (Note that when the browser 130B initiated, the parent window was already created in table 550.)

In step 930, the browser 130B executes the CCI event requested by the CCI redirector 145B. The local Web browser 130B has the same state, i.e. executes the same CCI Event 132A, as the remote Web browser 130A, but uses the mapping of the local system 150B to create the identical document view in both places. The result of the execution 930 of CCI event 132A is returned from the Web browser 130B to the CCI redirector 145B and may be stored and updated 940, if appropriate, by the CCI redirector 145B for later use in the table 550. Step 950 returns control to step 720 which continues to call process 800 until it terminates in step 730.

Messages and/or data are handled as Message Events 165A in a similar fashion as controls, i.e., CCI events 132.

Figure 10 is a flowchart of a process 1000, executed by both the sending 150A and receiving 150B machines, showing the steps of initialization, processing input messages, and termination done by Message redirector 155 (sender or receiver) during the Web collaboration session.

In a message-driven window system (for example, Windows 3.1 or OS/2), an application obtains data (parameter 634) from input devices 190 (mouse, keyboard, pen, etc.) in the form of messages (mouse message, keyboard message, pen message, etc.) that the operating system 195 sends to the application through its message queue 165.

The Message redirector 155 is used to implement the sharing of data from input devices 190 among clients 150. When a user 150A inputs some data into the Web document, these input data are transformed by the operating system 195 into messages which are kept in the message queue 165A. The Message redirector 155A (a source or sender) captures or hooks 191 these input data by intercepting the messages sent to the Web browser's objects (e.g., button object, text entry object, list box object) through the message queue 165.

To have the same input to the Web document at other clients 150B, the Message redirector 155A (a source or sender) sends the intercepted messages to its corresponding Message redirector 155B (a receiver) through the Web Sharing Manager 160B. Then, the receiver Message redirector 155B puts these messages into the message queue 165, on system 150B, in the same way as if these messages are generated by the operating system 195B for the local input devices 190B.

For example, if the user 150A types his/her name into the text entry object for name input in a Web document, the message redirector 155A intercepts this keyboard message and sends it to the corresponding message redirector 155B at other clients 150B. At each remote client 150B, the sent keyboard messages are put into the message queue 165, on machine 150B, and are retrieved by the text entry object for name input in the similar way as if these key strokes are locally input by the user 150B. As the result, all users will see the same name in their Web document.

As another example, when machine 150B receives a message event 165A, the message redirector 155B maps the window ordering number 520A,B to the window handle 530B so that the parameter 634 is placed in the message event queue 165B associated with the mapped window handle 530B.

Therefore, at the beginning of the Web collaboration, the Message redirector 155 uses a known feature provided by the operating system 195 to hook (1010, 191) the message queue 165.

In step 1020, the input message is processed by message redirector 155 as a sender (when the operating system 195 recognizes that the browser retrieved a

Message Event 165A from the message queue) or a receiver (when the Web Sharing Manager 160A passes the Communication Packet 600 to the message redirector 155) of input message. This step 1020 is elaborated for a message sender in Figure 11 and for a message receiver in Figure 12. When the Web collaboration is terminated, the message redirector 155, in step 1030, uses a known feature of the operating system to unregister message queue hooking.

In step 1030, the message redirector 155 terminates the processing by unregistering the hooking of the message event queue 165, i.e., if the browser 130 receives a message from the queue 165 the message redirector 155 will not be called and step 1020 will not execute.

Figure 11 is a flowchart of the steps 1020 implemented on the source or sending client 150A that intercepts the input message from the message queue 165, at a sending machine 150A. The message redirector 155A first determines 1110 that the machine 150 is the source/sender 150A of an input message intercepted from the message queue 165A. Based on the type 166 of the message, each message is checked to determined whether or not it is necessary to send the intercepted message to others clients 150B (step 1120). This step 1120 for processing each message is elaborated in Figure 14. If the message is needed to be sent to other clients 150B, the source message redirector 155A enters 1130 the type 632 and parameter 634 of the input message into the network web sharing data packet 630. (In general the type 632 is the message type 166 and the parameter 634 includes the fields 167, 168, and 169 in the queue entry 171.) The message redirector 155A sends 1135 the data packet 630 to its source Web Sharing Manager 160A. The source Web Sharing Manager 160A then creates 1140 the entire communications packet 600 by entering the receiver ID 622 and packet length 624 in the Web Sharing Manager header 620 and providing any other communication header 610 information required. The packet is then sent through the interface 120 by the Web Sharing Manager 160A using known techniques, e.g. TCP/IP. Step 1150 terminates the message process 1020 for the source machine 150A for any given input message from the message queue 165 and returns control to step 1020.

Figure 14 is a flowchart of the step 1120 performed by the message redirector 155. The operating system 195 retrieves an entry 171 from the message queue 165 (step 1410) and it checks if the hook 191 is enabled (step 1420). If the hook is disabled, the entry will be delivered to the destination 193. Since the message redirector 155 has hooked 191 the queue 165, the message director will receive all entries 171 and it examines if the message type is of interest as described above (for example, key code, mouse button click or double click, mouse move, scrolling message, tagged messages). If the message type 166 is a tagged message (step 1440) indicating that the message came from a source client 150A, the message redirector changes the message type 166 back by removing the tag (step 1450) and this entry 171 is passed to the destination 193. The message redirector 155 only duplicates the messages destined to the Web browser by examining the destination field 168 (step 1460). If the destination field 168 is the Web browser window or its child windows, the processing is continued 1470 to step 1130 of Figure 11.

Figure 12 is a flowchart of the step 1020 implemented on the client 150B that receives an input message from the Web Sharing Manager 160B. The message redirector 155B first determines that the machine 150 is the receiver 150B of a Web sharing data packet from the Web Sharing Manager 160B. The receiver message redirector 155B then retrieves the type 632 of the message and its parameters 634 from the data packet 630. (In general, the type 632 will be the message type 166 and the parameters 634 include the fields 167, 168, and 169 of the queue entry 171.) Since some parameters of the received message are only valid for the operating system where the message is intercepted, these parameters are converted to the local parameters of each client 150B by the message redirector 155B (step 1210). The massage redirector 155B puts the received message together with its converted parameters (step 1220) into the message queue 165, at the receiving machine 150B. The Web browser 130B retrieves 1225 the message from the message queue 165B and processes 1230 it in the same way as if this input message is from the local input devices 190B. The sender Web browser 130A and the receiver Web browser 130B then get the same input message, i.e. source user 150A and receiver user 150B share the same input data. Step 1240 terminates the message process 1020 for the receiver machine 150B for any given data packet 630 from the Web Sharing Manager 160B and returns control to step 1020.

Figure 13 is a drawing showing some preferred uses of the present system 100. In particular, an implementation 1300 of an interactive customer and agent collaboration, e.g. in a Home Banking system, using the web page sharing system 100 is described.

In the following scenario, a customer would like to open a new account with a bank advertising on the Internet. By using Interactive Home Banking prototype 1300, the bank agent 1320 is able to collaborate and help the customer 1310 to fill in the application form 1350, i.e. document 200, for a new account.

A customer uses a personal computer 150 to connect to the Internet 115 and runs a web browser (e.g. Netscape Navigator) 130 to browse the home page of a bank called "Interactive Home Bank." After reading the information on the products offered by the bank, the customer 1310 desires to open a new account with this bank, and therefore makes a data connection with the bank agent 1320 to establish a web collaboration session.

To facilitate the discussion, a voice connection 1365 may also be established through the telephone network

1360 or other mechanisms. The voice connection can be made through a standard direct dial telephone connection or can be made through the Internet via techniques 1368 described in U.S. patent application number 08/595,897 to Hortensius et al. entitled "Multipoint Simultaneous Voice and Data Services Using a Media Splitter Gateway Architecture" filed on February 6, 1996 (EP-A-789470).

In the web collaboration session, when a new web document 1350 is opened by one of the users (customer or bank agent), the same document will be displayed on the screen of another user. This sharing of browsing the web document 1350 is supported by the CCI redirector, as described above. For example, if the bank agent 1320 displays the application form 1350, the customer 1310 will see the same form of his/her screen. By using the functionality of the message redirector, the key strokes typed by the customer for the account name will also appear in the corresponding account name field on the bank customer's screen. (Here the account name 1321, comment 1302, signature 1303, submit 1304, and clear 1305 windows correspond to the child windows described in Figure 2). To facilitate the discussion, the customer 1310 and the bank agent 1320 may use the Annotation 142 and Remote Pointer 144 to mark and point to the areas on the application form. With this prototype, the customer and the bank agent can collaborate together and share the web document and the input data to fill in a form for opening a new bank account.

Many other applications are possible. For example, the document 1350 can be a contract, an order form, a menu, a design drawing, a listing of software code, a patent application, a bill of laden, or any other document 1350/200 generally used in commerce. The agent 1320 can be a sales person, a service provider (doctor, lawyer, accountant, etc.), designer (engineer, architect, programmer), manager, etc. The customer 1310 can be any person that would use the goods and/or services provided by the agent and/or any person the would work with (or for) the agent 1320 using or producing the document 1350. There can be more than one agent 1320 and/or more than one customer 1310. For example, the agent 1320 can be a teacher that teaches more that one student (customer) 1310. The customers 1310 can be a plurality of people in a single or multiple companies "conferencing" with multiple supplier company personnel and/or consultants (agents) 1320. Pen based interfaces permit all the collaborating parties to simultaneously sign documents 200, e.g. contracts (like rental car agreements), credit card statements, using the Internet without leaving their home or place of business.

Claims

1. A computer system (150A) having one or more memories (175), one or more central processing units (170), one or more displays, one or more user input devices (190), an operating system (195), and a network interface (120) with a network protocol for communicating to a network (115), the network connected to one or more servers (110) and one or more shared computers (150B), the computer further comprising:

   a browser (130) for requesting one or more pages from one or more of the servers by sending one or more browser requests (132A) though the network interface and over the network to the server and the browser receiving from the server one or more of the requested pages sent over the network through the network interface;
   one or more redirectors (145, 155) capable of recreating as redirected browser requests one or more of the browser requests necessary to create a web page on the display; and
   a web sharing manager (160) connected to the network interface, for receiving the redirected browser requests, and for sending the redirected browser requests over the network to one or more of the shared computers.

2. The computer system of claim 1, wherein the browser is capable of generating and processing one or more events and said operating system is capable of processing one or more messages, and said system further comprises a queue (165) for stacking one or more queued entries, one or more of the queued entries each containing one of the messages, and one or more of the queued entries each containing one of the events;
   and wherein said one or more redirectors are capable of monitoring and recreating as redirected queued entries one or more of the queued entries, and said web sharing manager is capable of receiving the redirected queued events from the redirector and incorporating each of the redirected queued events in a packet conforming to a network protocol which can be sent through the network interface and over the network to one or more of the shared computers.

3. The computer system of claim 2, where one or more of the messages include any one of the following: an operating system message, an application message, a input device message, and a message produced by one or more of the shared computers.

4. The computer system of claim 3, where one or more of the messages are created by one or more of the following: keystrokes from a keyboard, mouse clicks and movements, pen input, speech signals from a speech recognition system, and operating system messages.

5. The computer system of claim 2, 3, or 4, where one or more of the events is a browser request.

6. The computer system of any of claims 2 to 5, where one or more of the queued events are Common Client Interface (CCI) events created by the browser.

7. The computer system of claim 6, where the CCI event is converted into an operating system protocol, where the operating system protocol is Dynamic Data Exchange (DDE).

8. The computer system of claim 6 or 7, where the CCI event includes any one or more of the following: a request for opening a page, a request to change a window size, a request to register a protocol, a request to unregister a protocol, a request to get window information, a request to be notified when a new page is open, and a request to be notified when a window size is changed.

9. The computer system of any of claims 2 to 9, where the sent packets (600) have a communication header (610) with information used by the network protocol and one or more Web Sharing Manager Headers (620), each Web Sharing Manager Header having a Receiver ID field (622) and a Packet Length field (624), the former having a receiver value that identifies the redirector and the latter having a value specifying the length of Web Sharing Manager Data (630), the Web Sharing Manager Data having a Data Type field (632) identifying the type of the event and a Parameter field (634) identifying the parameters associated with the event.

10. The computer system of any preceding claim, wherein the browser is capable of displaying one or more of the requested pages on the display with one or more messages of information from the user input device; wherein said one or more redirectors are capable of recreating one or more of the messages as redirected messages; and wherein the web sharing manager is capable of receiving the redirected messages, and sending the redirected messages over the network to one or more of the shared computers, so that the shared computer displays a copy of the page on a shared display.

11. The computer system of any preceding claim, where the server is an HTML server and one or more of the pages is an HTML page.

12. The computer system of any preceding claim, where the input device is a pen capable of producing a signature and the document has one or more windows where the signature is placed.

13. A method of operating a computer system (150A) having a display, a network interface (120) for communicating to a network (115), and a browser (130) for sending requests for pages over the network and displaying the requested pages on the display, said method including the steps of:

retrieving a message (171) from a memory;
duplicating the message if the message is necessary for a browser to create a page on the display; and
sending the message over a network to a sharing computer (150B) so that a receiving browser on the sharing computer can use the message to create a received page on a received display that is identical to the page.

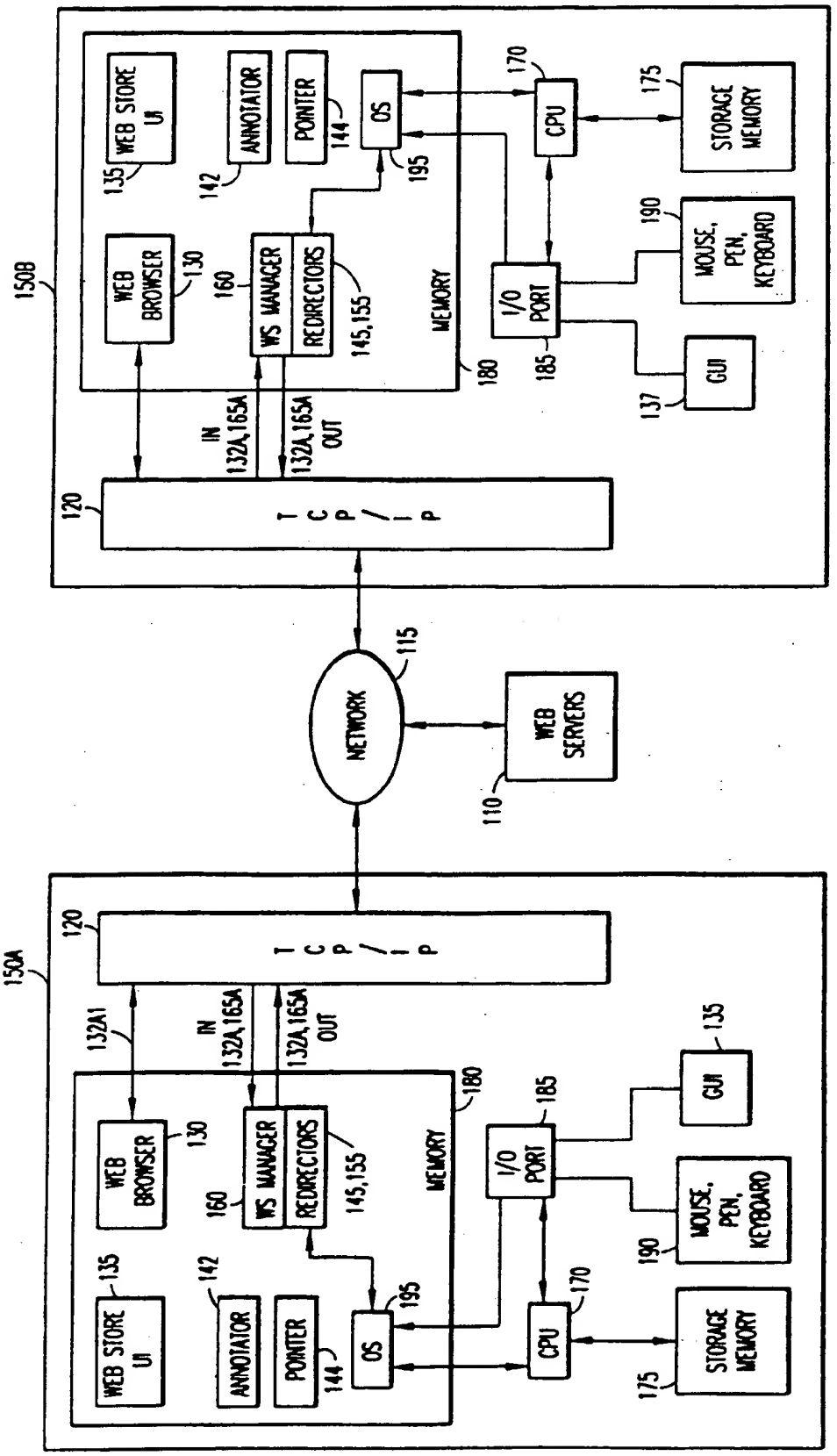
150B
WEB STORE UI
135
ANNOTATOR
142
POINTER
144
OS
195
CPU
170
STORAGE MEMORY
175
WEB BROWSER
130
160
WS MANAGER
REDIRECTORS
145,155
MEMORY
180
I/O PORT
185
MOUSE, PEN, KEYBOARD
190
GUI
137
IN 132A,165A
OUT 132A,165A
120
T C P / I P
NETWORK
115
WEB SERVERS
110
100
150A
132A1
GUI
135

FIG. 1

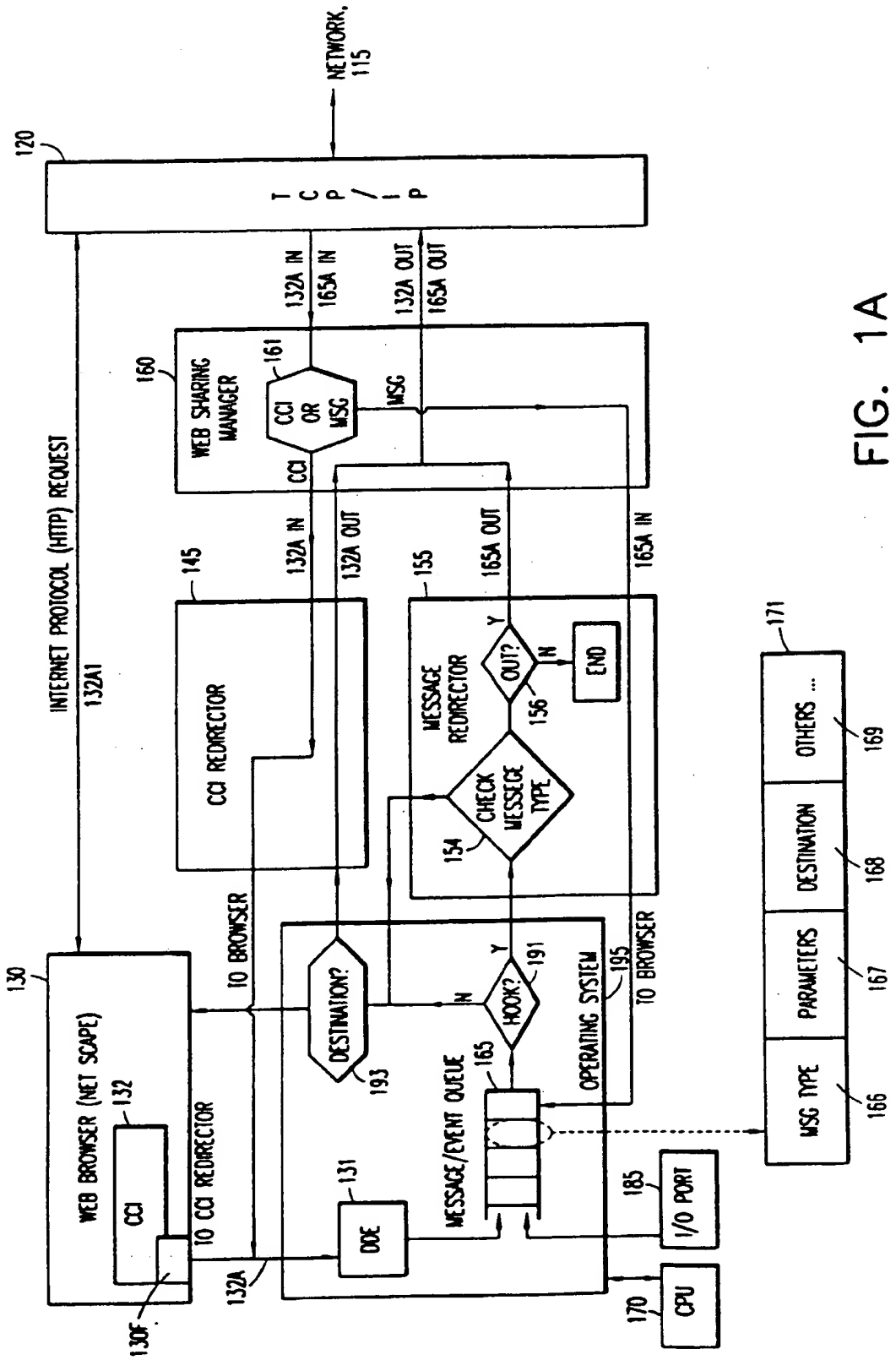
NETWORK, 115
120
T C P / I P
INTERNET PROTOCOL (HTTP) REQUEST
132A1
132A IN
165A IN
132A OUT
165A OUT
160
WEB SHARING MANAGER
161
CCI OR MSG
MSG
CCI
145
CCI REDIRECTOR
155
MESSAGE REDIRECTOR
OUT?
Y
N
END
156
CHECK MESSAGE TYPE
154
130
WEB BROWSER (NET SCAPE)
132
CCI
130F
IO CCI REDIRECTOR
IO BROWSER
DESTINATION?
193
HOOK?
191
OPERATING SYSTEM
195
MESSAGE/EVENT QUEUE
165
131
DDE
132A
185
I/O PORT
170
CPU
171
OTHERS ...
169
DESTINATION
168
PARAMETERS
167
MSG TYPE
166

FIG. 1A

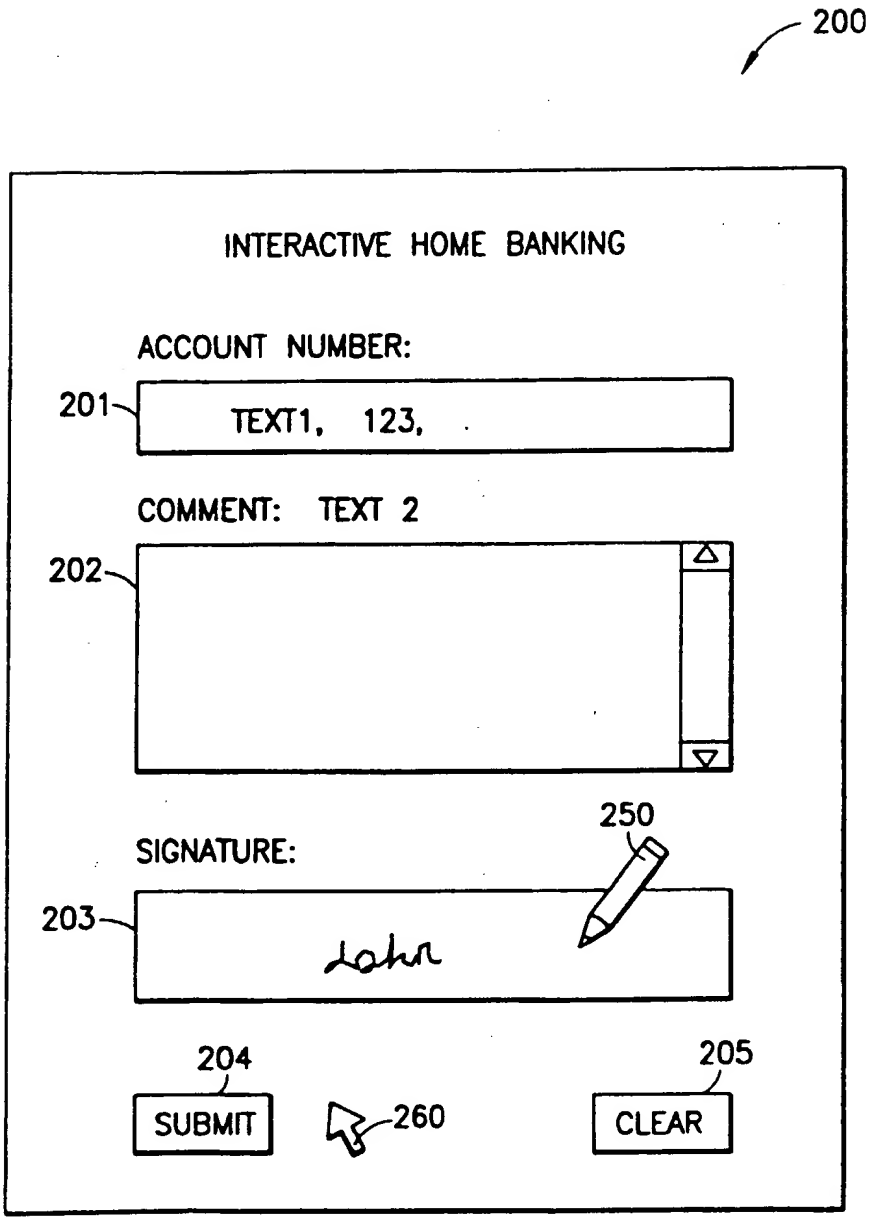
200
INTERACTIVE HOME BANKING
ACCOUNT NUMBER:
201
TEXT1, 123,
COMMENT: TEXT 2
202
250
SIGNATURE:
203
John
204
205
SUBMIT
260
CLEAR

FIG. 2

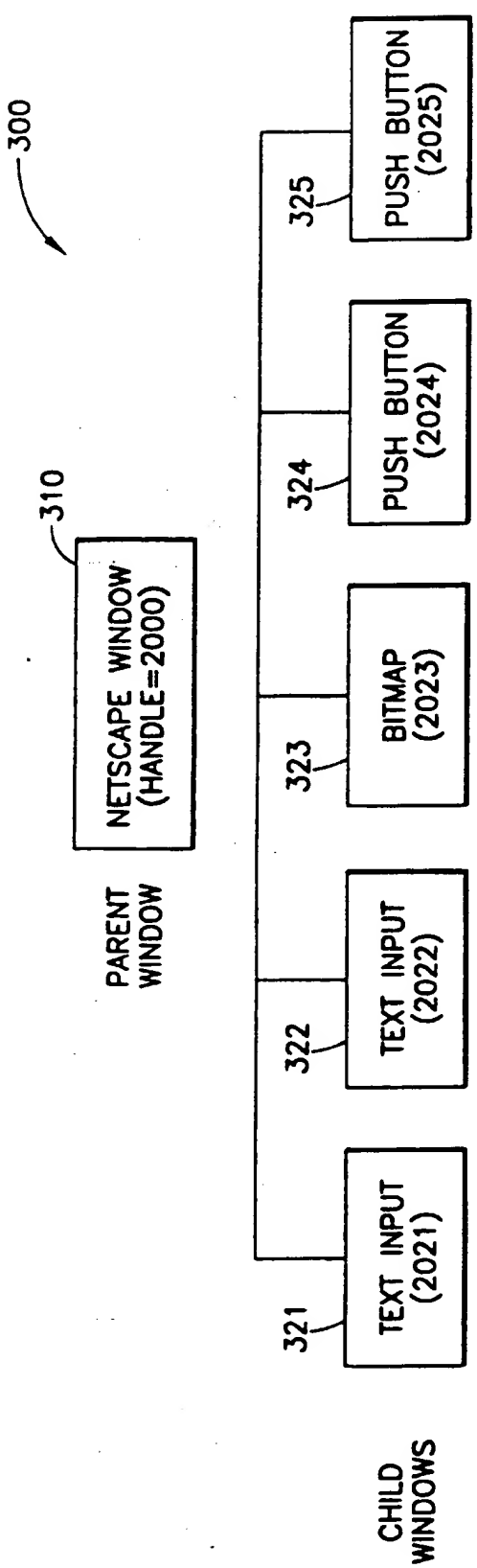
300
310
NETSCAPE WINDOW (HANDLE=2000)
PARENT WINDOW
321
TEXT INPUT (2021)
322
TEXT INPUT (2022)
323
BITMAP (2023)
324
PUSH BUTTON (2024)
325
PUSH BUTTON (2025)
CHILD WINDOWS

FIG. 3

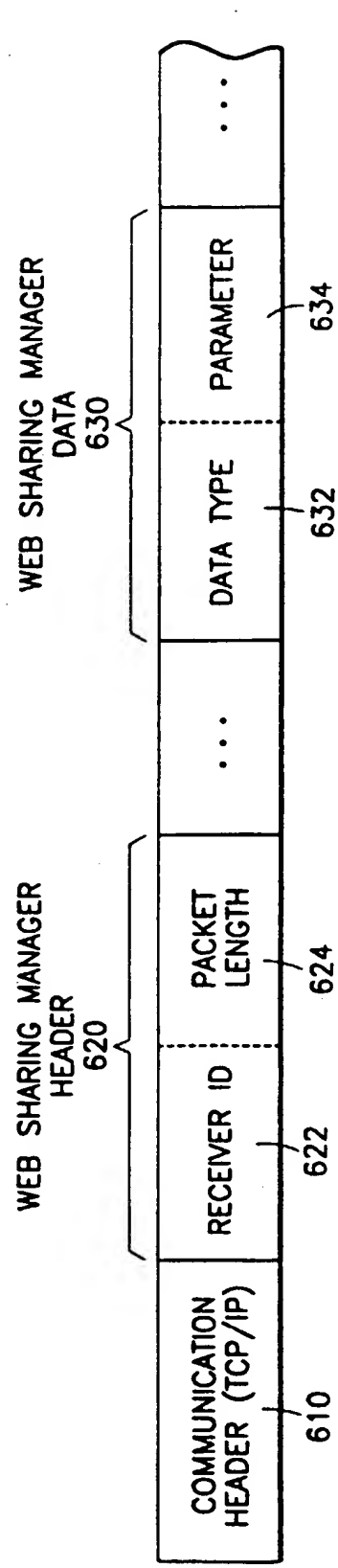
600
WEB SHARING MANAGER HEADER
620
WEB SHARING MANAGER DATA
630
COMMUNICATION HEADER (TCP/IP)
610
RECEIVER ID
622
PACKET LENGTH
624
...
DATA TYPE
632
PARAMETER
634
...

FIG. 6

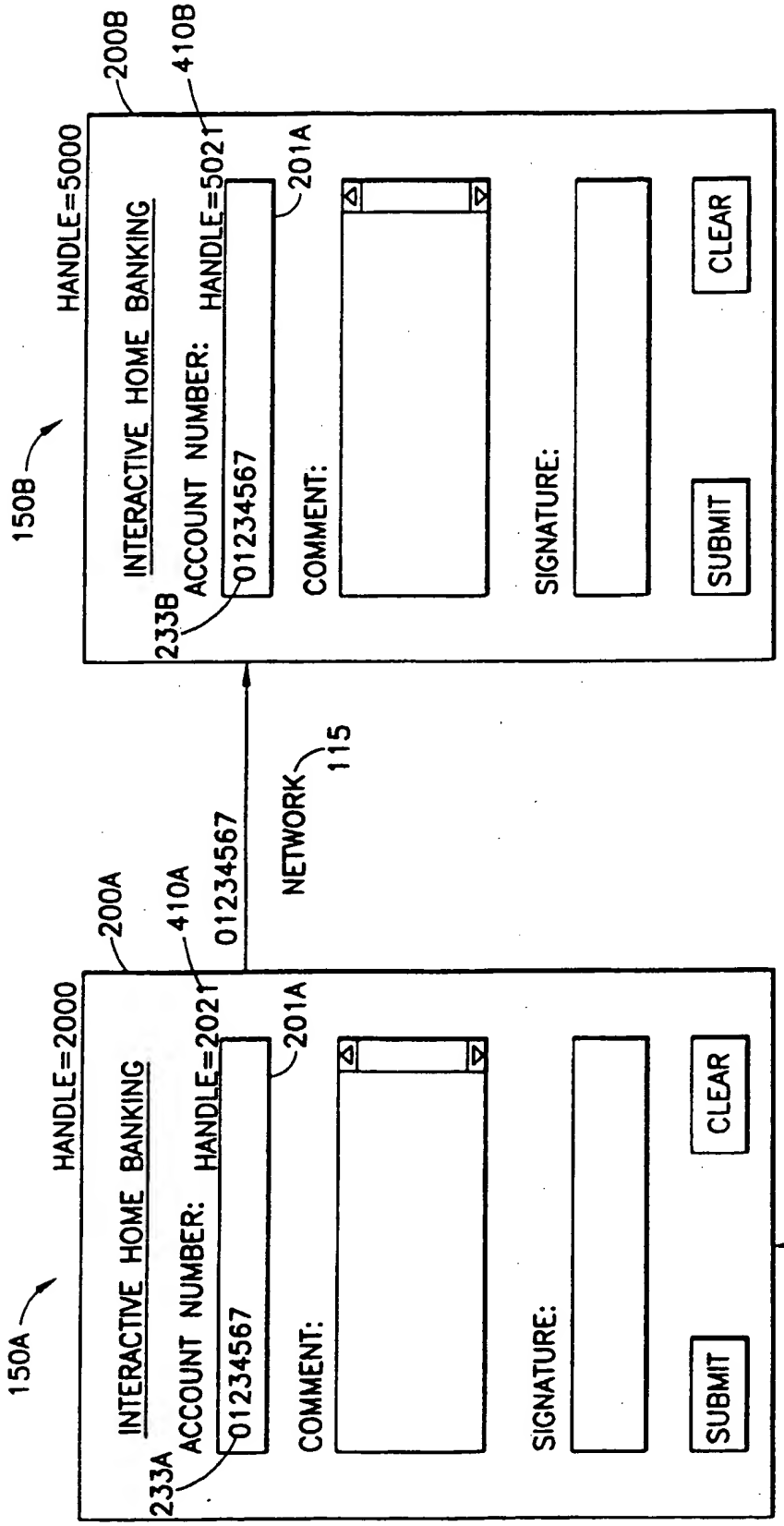
150A
HANDLE=2000
INTERACTIVE HOME BANKING
ACCOUNT NUMBER:
HANDLE=2021
01234567
233A
COMMENT:
SIGNATURE:
SUBMIT
CLEAR
200A
410A
201A
01234567
NETWORK
115
150B
HANDLE=5000
INTERACTIVE HOME BANKING
ACCOUNT NUMBER:
HANDLE=5021
01234567
233B
COMMENT:
SIGNATURE:
SUBMIT
CLEAR
200B
410B
201A
BANK AGENT MACHINE
01234567
KEYBOARD
190
CUSTOMER MACHINE

FIG. 4

500

510

520A ORDERING NUMBER	530A WINDOW HANDLE	540A WINDOW TYPE
0	2000	WEB BROWSER WINDOW
1	2021	TEXT INPUT
2	2022	TEXT INPUT
3	2023	BITMAP
4	2024	PUSH BUTTON
5	2025	PUSH BUTTON
...	...	...

FIG. 5A

550

520B ORDERING NUMBER	530B WINDOW HANDLE	540B WINDOW TYPE
0	5000	WEB BROWSER WINDOW
1	5021	TEXT INPUT
2	5022	TEXT INPUT
3	5023	BITMAP
4	5024	PUSH BUTTON
5	5025	PUSH BUTTON
...	...	...

FIG. 5B

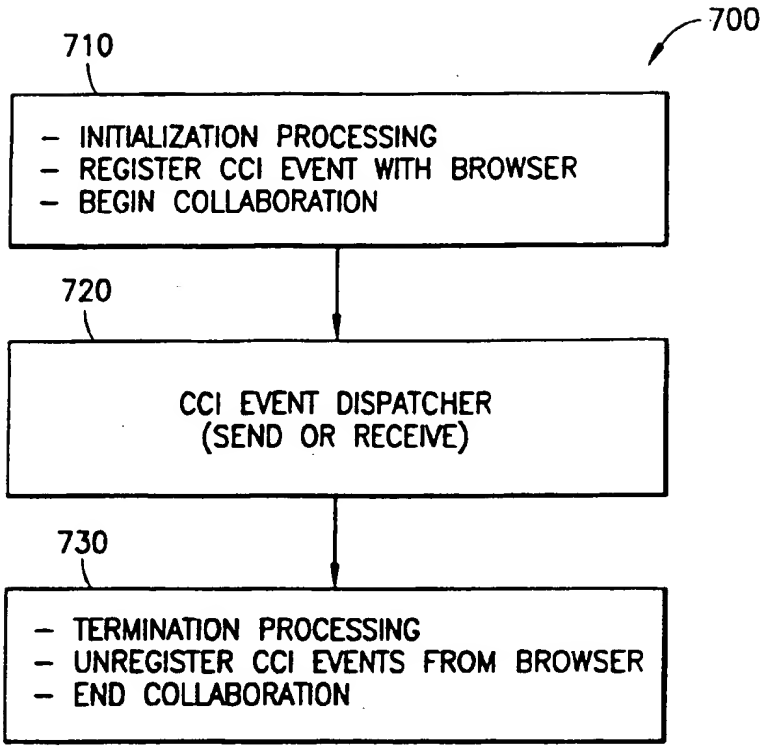
700
710
– INITIALIZATION PROCESSING
– REGISTER CCI EVENT WITH BROWSER
– BEGIN COLLABORATION
720
CCI EVENT DISPATCHER
(SEND OR RECEIVE)
730
– TERMINATION PROCESSING
– UNREGISTER CCI EVENTS FROM BROWSER
– END COLLABORATION

FIG. 7

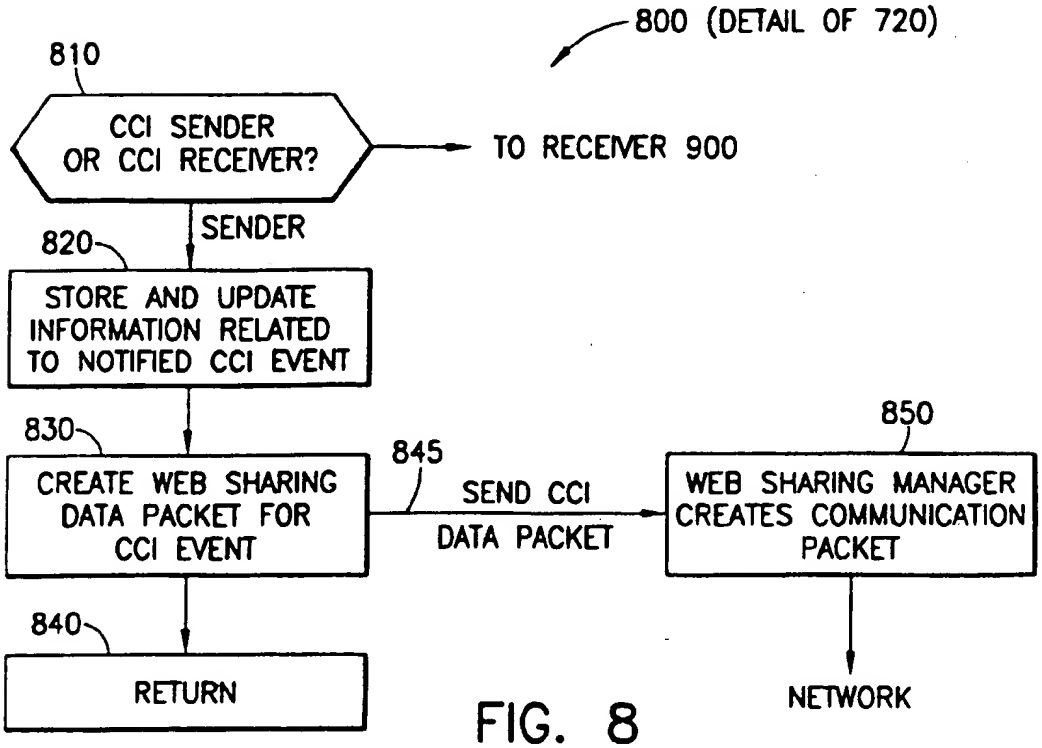
800 (DETAIL OF 720)
810
CCI SENDER
OR CCI RECEIVER?
TO RECEIVER 900
SENDER
820
STORE AND UPDATE
INFORMATION RELATED
TO NOTIFIED CCI EVENT
830
CREATE WEB SHARING
DATA PACKET FOR
CCI EVENT
845
SEND CCI
DATA PACKET
850
WEB SHARING MANAGER
CREATES COMMUNICATION
PACKET
840
RETURN
NETWORK

FIG. 8

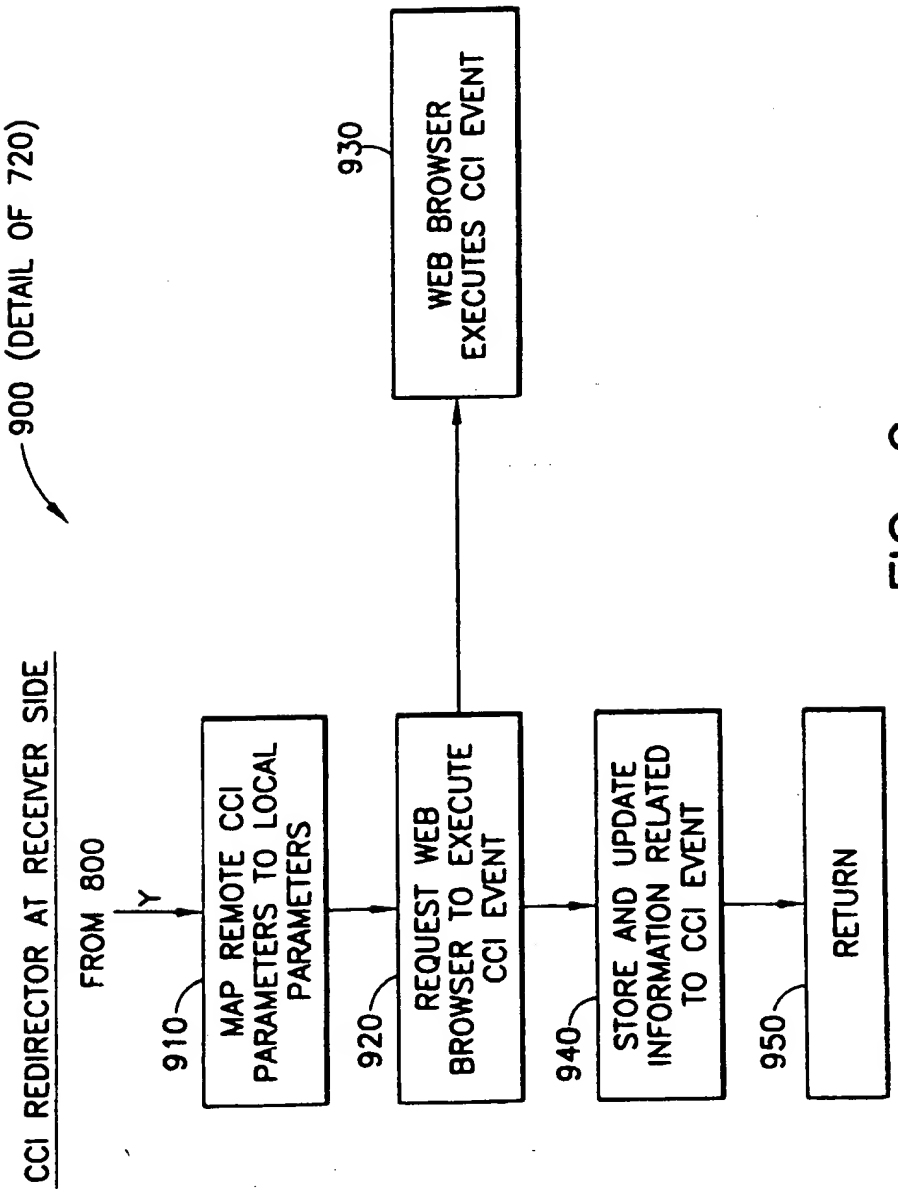
CCI REDIRECTOR AT RECEIVER SIDE
900 (DETAIL OF 720)
FROM 800
Y
910
MAP REMOTE CCI PARAMETERS TO LOCAL PARAMETERS
920
REQUEST WEB BROWSER TO EXECUTE CCI EVENT
930
WEB BROWSER EXECUTES CCI EVENT
940
STORE AND UPDATE INFORMATION RELATED TO CCI EVENT
950
RETURN

FIG. 9

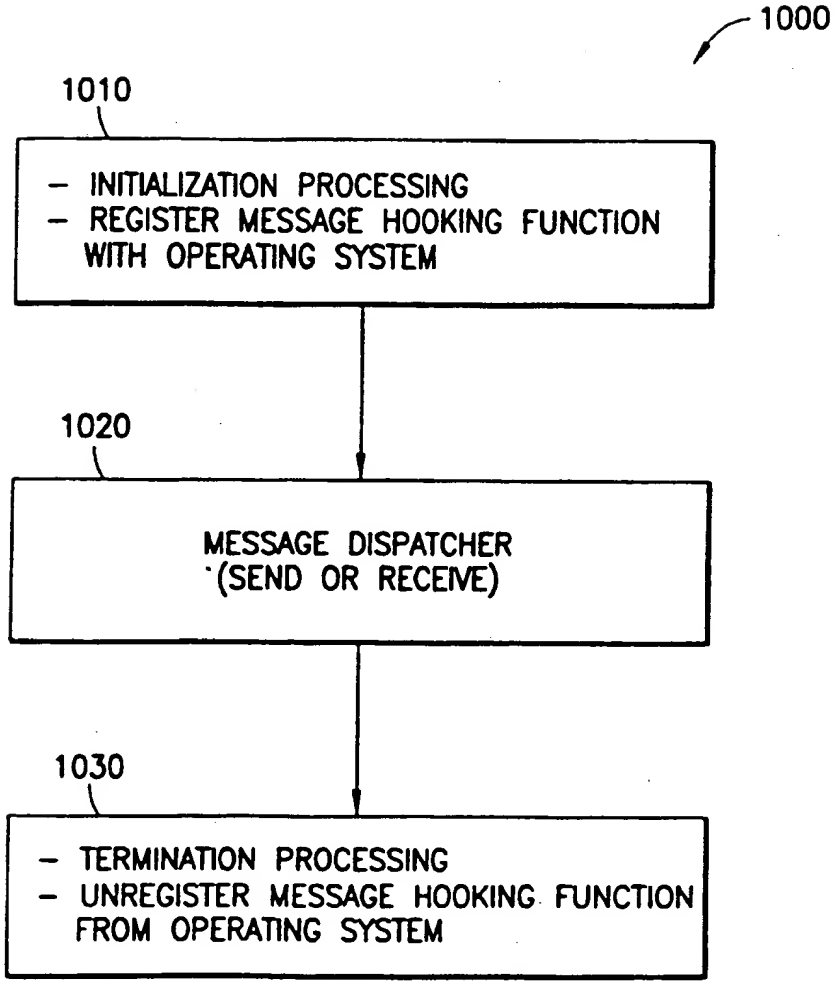
1000
1010
– INITIALIZATION PROCESSING
– REGISTER MESSAGE HOOKING FUNCTION WITH OPERATING SYSTEM
1020
MESSAGE DISPATCHER (SEND OR RECEIVE)
1030
– TERMINATION PROCESSING
– UNREGISTER MESSAGE HOOKING FUNCTION FROM OPERATING SYSTEM

FIG. 10

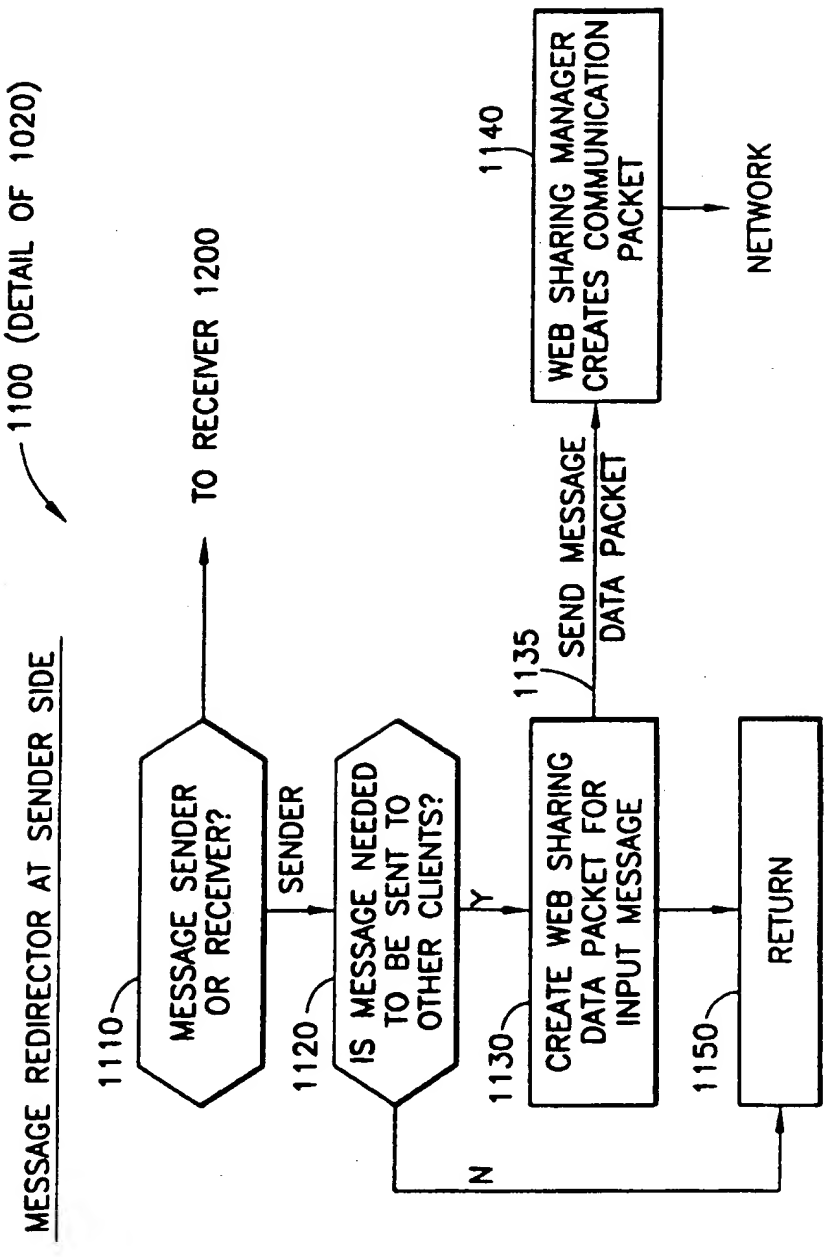
MESSAGE REDIRECTOR AT SENDER SIDE
1100 (DETAIL OF 1020)
MESSAGE SENDER OR RECEIVER?
1110
TO RECEIVER 1200
SENDER
IS MESSAGE NEEDED TO BE SENT TO OTHER CLIENTS?
1120
Y
N
CREATE WEB SHARING DATA PACKET FOR INPUT MESSAGE
1130
1135
SEND MESSAGE DATA PACKET
WEB SHARING MANAGER CREATES COMMUNICATION PACKET
1140
NETWORK
RETURN
1150

FIG. 11

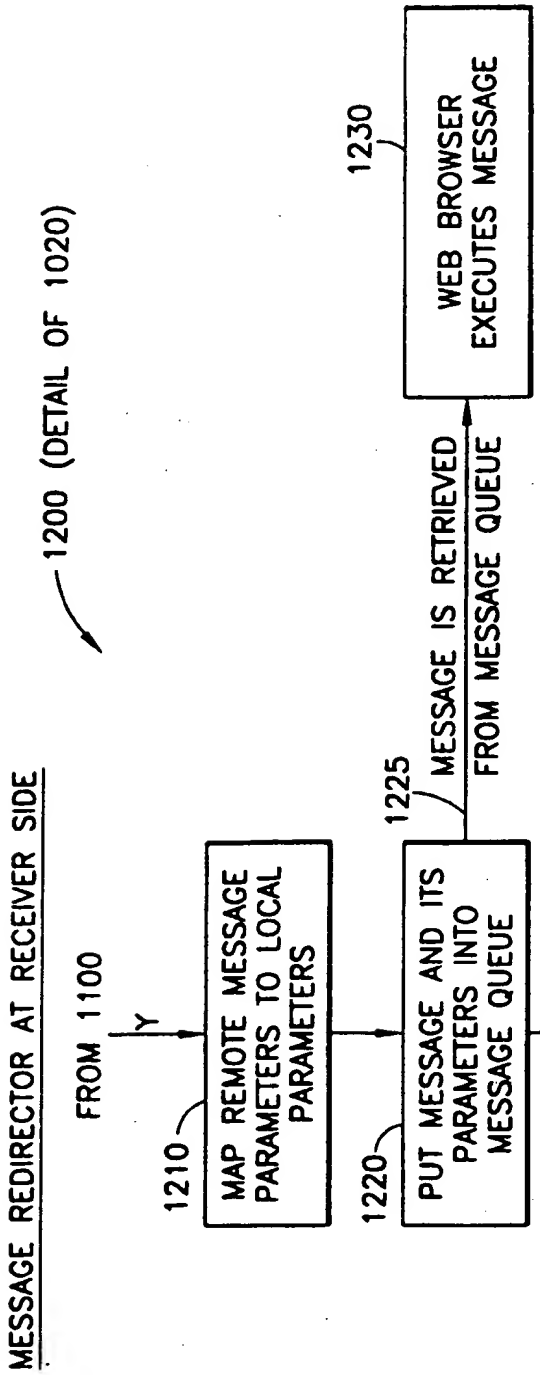
MESSAGE REDIRECTOR AT RECEIVER SIDE
1200 (DETAIL OF 1020)
FROM 1100
Y
1210
MAP REMOTE MESSAGE PARAMETERS TO LOCAL PARAMETERS
1220
PUT MESSAGE AND ITS PARAMETERS INTO MESSAGE QUEUE
1225
MESSAGE IS RETRIEVED FROM MESSAGE QUEUE
1230
WEB BROWSER EXECUTES MESSAGE
1240
RETURN

FIG. 12

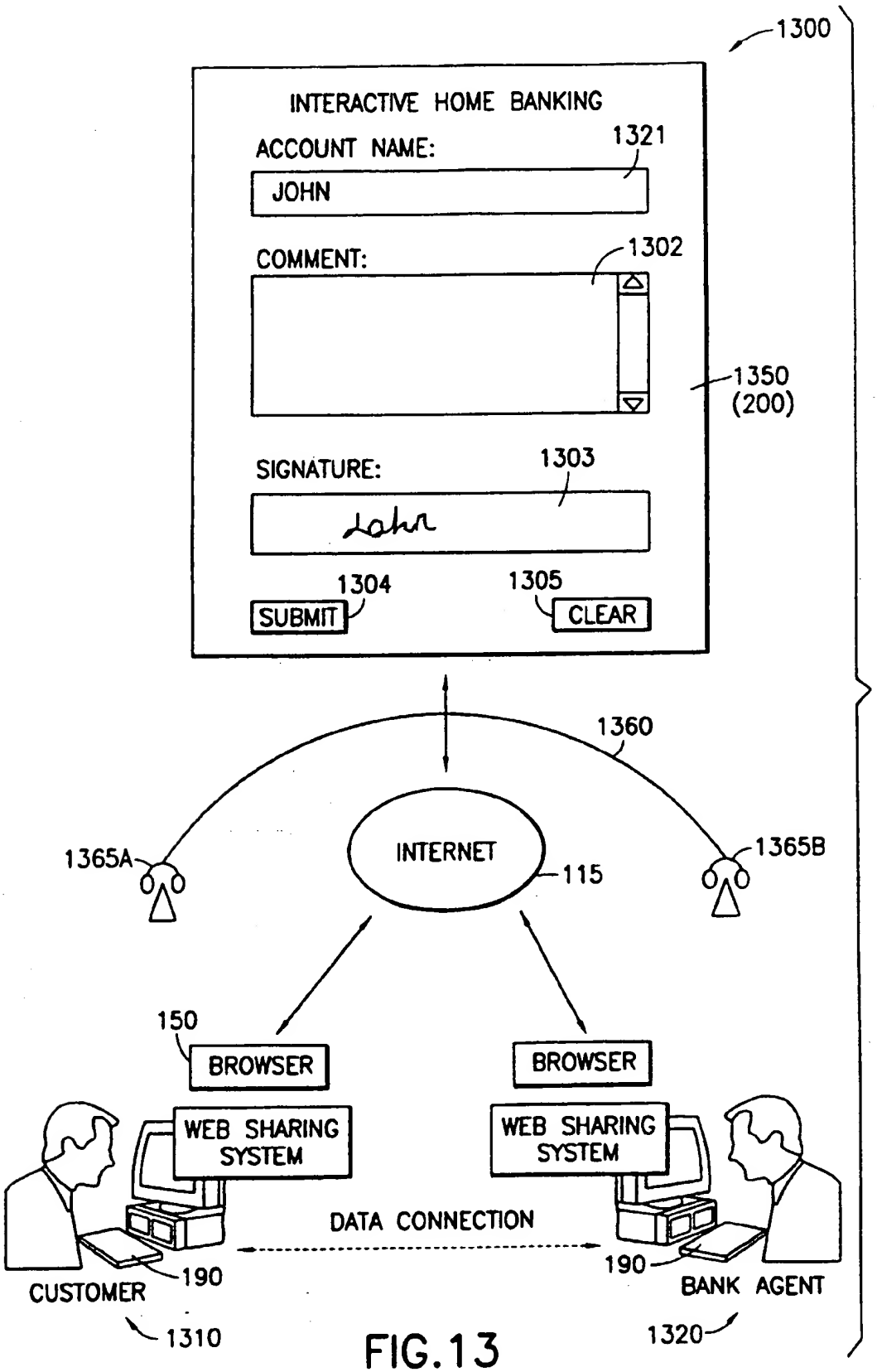
1300
INTERACTIVE HOME BANKING
ACCOUNT NAME:
1321
JOHN
COMMENT:
1302
1350
(200)
SIGNATURE:
1303
1304
SUBMIT
1305
CLEAR
1360
INTERNET
115
1365A
1365B
150
BROWSER
BROWSER
WEB SHARING SYSTEM
WEB SHARING SYSTEM
DATA CONNECTION
190
190
CUSTOMER
BANK AGENT
1310
1320

FIG.13

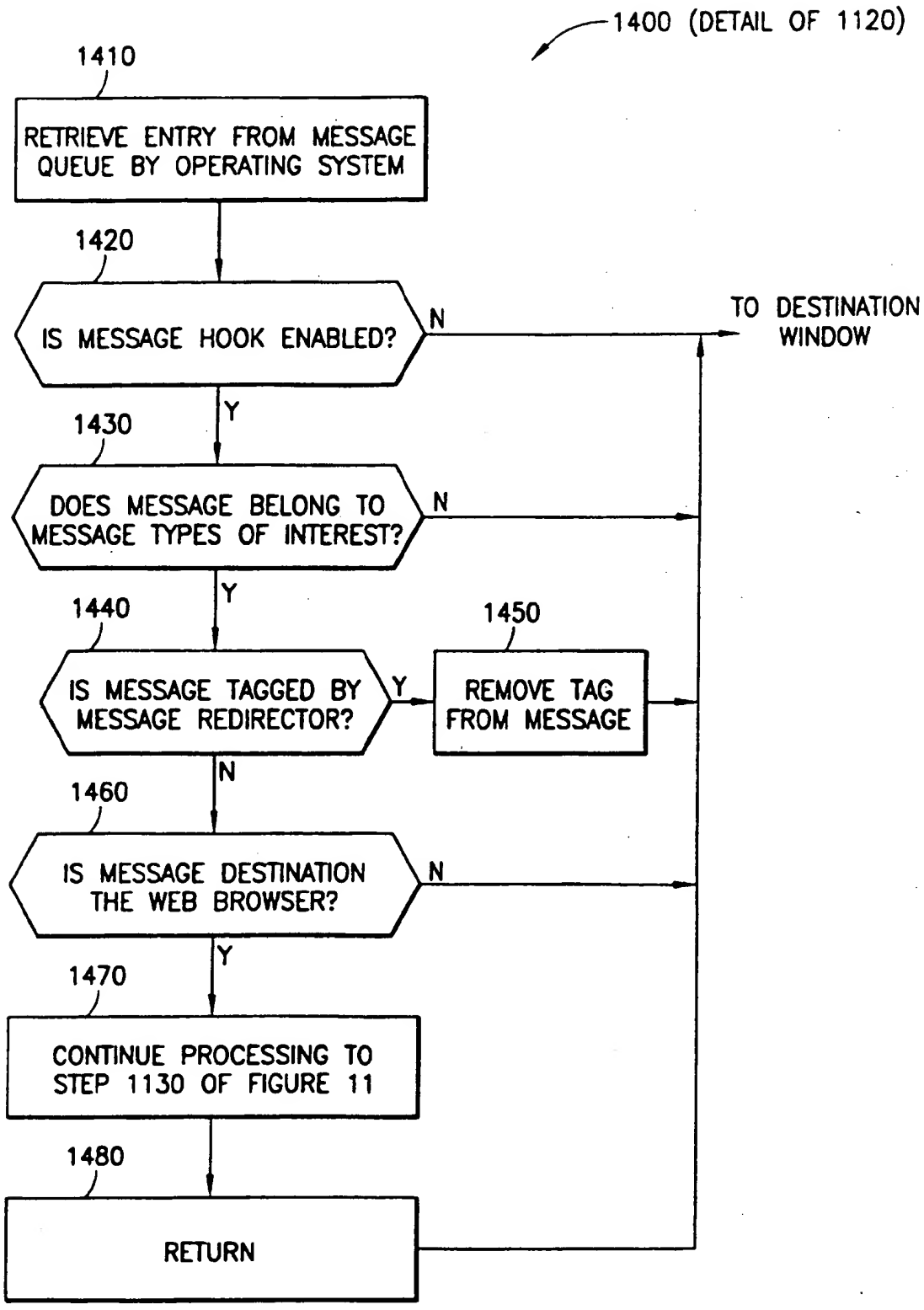
1400 (DETAIL OF 1120)
1410
RETRIEVE ENTRY FROM MESSAGE QUEUE BY OPERATING SYSTEM
1420
IS MESSAGE HOOK ENABLED?
N
TO DESTINATION WINDOW
Y
1430
DOES MESSAGE BELONG TO MESSAGE TYPES OF INTEREST?
N
Y
1440
IS MESSAGE TAGGED BY MESSAGE REDIRECTOR?
Y
1450
REMOVE TAG FROM MESSAGE
N
1460
IS MESSAGE DESTINATION THE WEB BROWSER?
N
Y
1470
CONTINUE PROCESSING TO STEP 1130 OF FIGURE 11
1480
RETURN

FIG. 14